# THE TEXAS INSTRUMENTS HOME COMPUTER

# IDEA BOOK

INCLUDES 50 READY-TO-RUN EDUCATIONAL PROGRAMS

DAVID H. AHL

# The Texas Instruments Home Computer Ideabook

# The Texas Instruments Home Computer Ideabook

## Includes 50 Ready-to-Run Programs

David H. Ahl

Illustrations: Wayne Kaneshiro

Creative Computing Press
Morris Plains, New Jersey

Library of Congress Number 83-71195
ISBN 0-916688-51-8

Printed in the United States of America
First printing April 1983
10 9 8 7 6 5 4 3 2 1

Creative Computing Press
39 E. Hanover Avenue
Morris Plains, NJ 07950

Dedication: To Betsy, for her friendship, encouragement and understanding.

## About the Author

David H. Ahl has a BEE from Cornell University, MBA from Carnegie-Mellon University and has done further work in educational psychology at the University of Pittsburgh.

He served in the Army Security Agency, was a consultant with Management Science Associates and a senior research fellow with Educational Systems Research Institute.

In early 1970, he joined Digital Equipment Corporation. As education product line manager, he formulated the concept of an educational computer system consisting of hardware, software and courseware and helped guide DEC into a leading position in the education market.

Mr. Ahl joined AT&T in 1974 as education marketing manager and was later promoted to manager of marketing communications for the unit later to become American Bell. Concurrent with this move, he started *Creative Computing* as a hobby in late 1974.

As *Creative Computing* grew, Mr. Ahl left AT&T in 1978 to devote full time to it. *Creative Computing* magazine today is Number 1 in software and applications. In January 1980, Ahl founded *SYNC* magazine and, over the years, has acquired or started several other publications.

Mr. Ahl is the author or editor of 16 books and over 150 articles about the use of computers. He is a frequent lecturer and workshop leader at educational and professional conferences.

# Contents

# Preface

Although Charles Babbage laid down several ideas for computing "engines," the forerunners of today's computers were largely developed in the early 40's as part of the war effort. The Robinson series cryptoanalytic machines developed in England in 1941 spawned many families of computers still in use today. The MIT differential analyzer and real-time aircraft simulation project led to the Whirlwind, and eventually to the immensely successful DEC family of PDP computers (Programmed Data Processors). And, of course, Eniac built at the University of Pennsylvania, was the guiding light behind Univac, IBM and many other successful manufacturers.

Many people today poke fun at these early machines and regard them as dinosaur-like relics. However, it is interesting to consider that a large-scale computer of about 25 to 30 years ago had about the same amount of power as a typical personal computer of today. It was generally not as reliable or user-friendly as a personal computer, and, of course, cost tens of thousands times as much. Why bring this up?

Because a computer of the 50's required that the programmer be very clever and resourceful to solve problems within the capabilities of the computer. He did not have vast gobs of memory available, blinding quick calculation speed, or random disk access. In other words, he had about the same problem to face as you do with your personal computer.

I do not mean to imply that your personal computer is not a full-fledged computer. It certainly is just as much a computer as a room-filling giant of today. However, because of the relatively small memory, it cannot store a large data base. Nor is it suitable for extensive word processing or massive calculations. Highly detailed graphics are best left to other machines as well.

What can we learn from the computing pioneers of the 50's that will help us today? Perhaps most important is the discipline of thoroughly analyzing a problem, breaking it down into manageable steps, and solving it a step at a time. It is also important to determine what can be done "off line" and what must be done on the computer.

That is what this book is all about. While it has more than 50 ready-to-run programs, the main thing you should look for from the book is an approach to solving problems—big and small. Some of the problems demonstrate the capability of the computer; others identify its shortcomings. It is important to be familiar with both the strengths and weaknesses of your tools so you can recognize the types of jobs for which they are suitable (and not suitable).

This book focuses primarily on mathematical and educational applications for the computer. There are many other excellent sources of information about other applications and for making best use of your personal computer. Using the approaches described in this book should enable you to easily convert programs and use applications from other books and magazines such as *Creative Computing.*

This book is designed to be read with a working computer at hand. While there is textual material to be read, the most important things are the experiments and problems to be tried with your own computer. The book raises many questions for which you should try to find answers. There are no answers to these questions and problems in the back of the book; you should be able to discover the answers as you work the problems out on your computer.

You will be able to incorporate many of the routines and approaches in the book into programs of your own as you use your computer to deal with "real world" problems. Other programs will simply point you in the right direction. And some of the programs in the book are just plain fun. Learn. Experiment. Have fun!

Morristown, New Jersey
March 1983

David H. Ahl

# The Texas Instruments Home Computer Ideabook

# 1

# Drill and Practice

Throughout life, there are certain things that simply must be memorized. Obvious things that fall into this category are the addition and multiplication tables, the spelling and meaning of words, how to tell time, and the monetary system.

However, depending upon one's chosen profession, there are many other things to be memorized. A doctor must know what diseases match what symptoms. A chemist must know the gas laws, the properties of elements and so on. A pilot must instantaneously know the meaning of readings on scores of instruments.

To memorize a set of facts, you must go over them again and again and keep trying different variations. Here is where the computer comes in. It is able to present randomly scores of different problems to you for as long as you wish. Some programs will automatically adjust to your level of competence and will grade you; other programs simply present the problems and leave the grading up to you.

There are four programs in this chapter and two in the Science chapter which present material in a drill and practice format. Examine the methods used in these programs and then make up some drill programs of your own for subjects with which you are having trouble, or make up programs for other members of your family.

# Addition Practice

This program demonstrates a simplified addition drill and practice routine. This type of drill is sometimes called computer assisted instruction (CAI), although CAI can also apply to tutorial and other approaches as well.

When the program is run, it will first ask "No. of digits?" You enter a number and each addend will contain that many or fewer digits.

The program will present any number of practice problems that you specify. The program presents each problem in turn. The program will not proceed to the next problem until the current one has been answered correctly. After the last problem is answered correctly, the score is printed with an appropriate comment.

There are many improvements and extensions possible in a program like this one. For example, you might want to modify the program so it tells the user the correct answer after a problem has been answered incorrectly two (or three) times.

A more complicated modification would be to change the program to present different kinds of arithmetic problems such as subtraction, multiplication, and division.

Some of these modifications have been made in the next program.

```
10 RANDOMIZE
20 PRINT "ADDITION PRACTICE"
::
30 INPUT "NO. OF DIGITS =":N
40 INPUT "NO. OF PROBLEMS ="
:P
50 W=0
60 X=X+1
70 Q=0
80 A=INT(10^N*RND)+1
90 B=INT(10^N*RND)+1
100 PRINT
110 PRINT TAB(8-INT(LOG(A)/L
OG(10)+1));A
120 PRINT TAB(7-INT(LOG(B)/L
OG(10)+1));"+";B
130 PRINT "  --------"
140 C=A+B
150 PRINT TAB(7-INT(LOG(C)/L
OG(10)+1));
160 INPUT G
170 IF G=C THEN 230
180 Q=Q+1
```

```
190  IF Q>1 THEN 210
200  W=W+1
210  PRINT "WHAT? TRY AGAIN"
220  GOTO 100
230  PRINT "RIGHT!";
240  IF X>=P THEN 270
250  PRINT "HERE IS ANOTHER O
NE."
260  GOTO 60
270  PRINT ::"YOU GOT";P-W;"C
ORRECT THE":"FIRST TIME."
280  PRINT
290  IF W<.22*P THEN 320
300  PRINT "BUT YOU MISSED";W
310  STOP
320  PRINT "GOOD WORK!"
330  STOP

>RUN
ADDITION PRACTICE

NO. OF DIGITS =3
NO. OF PROBLEMS =5

      731
    + 761
  -------
   ? 1492
RIGHT!HERE IS ANOTHER ONE.

      387
    +  28
  -------
    ? 415
RIGHT!HERE IS ANOTHER ONE.

      133
    + 408
  -------
    ? 541
RIGHT!HERE IS ANOTHER ONE.

      933
    + 987
  -------
   ? 1810
WHAT? TRY AGAIN

      933
    + 987
  -------
   ? 1920
RIGHT!HERE IS ANOTHER ONE.

      229
    + 103
  -------
    ? 332
RIGHT!

YOU GOT 4 CORRECT THE
FIRST TIME.

GOOD WORK!
```

# Addition Practice, Adjusted by Grade Level

One of the major disadvantages with many drill and practice exercises is that they tend to be either boring or frustrating, depending upon the ability of the user relative to the level of the material. To compensate for this, a method is needed which will adjust the difficulty of the problems to the ability of the user.

Ideally, such a system would weigh the most recent performance most heavily but would not ignore previous performance. It should allow a user to advance to more difficult problems than his current mastery level. It should also continue to give some practice on problems already mastered.

Some commercial software packages approach these goals along traditional lines, i.e., determine in which type of problems a student should receive practice by using a complicated computer managed instruction score recording and adjustment system.

The approach here is more innovative; it uses a single measure for each type of problem—call it "estimated grade level"—which meets all of the objectives stated above.

How does it work? The most recent problem presented counts 10% of the overall score if it was answered correctly and was over the current user grade level, or if it was answered incorrectly and was under the current user grade level. Otherwise it is ignored. This may be easier to visualize in the form of a chart:

| | | *Answer* | |
|---|---|---|---|
| | | *Right* | *Wrong* |
| *Problem* | *Higher than grade level* | Raise student grade level | Ignore |
| | *Lower than grade level* | Ignore | Lower student grade level |

At first glance this might look complex and somewhat goofy, however, what it really means is that a student is rewarded for doing a problem beyond his grade level but he is not penalized if he cannot do it. On the other hand he is penalized if he cannot do a problem lower than his grade level, but is not rewarded for doing one lower.

Each problem affects the estimated grade level a little bit, with the most recent problems being weighed the most heavily. If the current grade level of a student is L and the level of the most recent problem to be averaged in is **P**,

then the averaging formula is simply:

$$L = .9L + .1P$$

The remaining task before a program can be written is to assign a grade level to each problem presented. Unfortunately, this will vary depending upon the local school system, the textbook used, and the teaching method. Also a huge data base can not be stored in a small computer, so it is desirable to devise a simple method of determining grade level for different problems. One straightforward approach is to present problems up to one-half a grade level over and under where the student currently is. Thus the overall range of problems for a student at grade level 3.2 would be 2.7 to 3.7.

How do we generate the right problems? Consider one type of skill, vertical addition. It is normally introduced in the first grade and continues through Grade 4 (actually 4.9). The simplest problem in this program is 1 + 1 and the most difficult is 999 + 999. Since learning is not a linear process (it is slow at first, and then progresses rapidly), an exponential formula can be used. For example:

$$\text{Addend} = 1.73 \times (\text{Grade level})^4$$

or

$$\text{Grade level} = \sqrt[4]{\text{Addend}/1.73}$$

This means that students at various grade levels will be working with the following maximum addends:

| *Grade level* | *Addend* |
|---|---|
| 1.0 | 1 |
| 2.0 | 27 |
| 3.0 | 140 |
| 4.0 | 442 |
| 4.9 | 997 |

Now it is a relatively straightforward, although somewhat tedious, matter to tie all these elements together in a computer program.

A few notes about the program. The variable G2 is the problem grade level that is always within one-half of a grade level of the current student level, G1. The complicated mess in Statement 340 produces a moving grade level average.

The recording of the grade level and carrying it over to the next lesson is a manual process. On a computer system with a permanent mass storage device, this would be kept on the system.

There are many possible changes and extensions to this program. For example it could present different types of problems such as horizontal addition, vertical and horizontal subtraction, as well as multiplication, division and fraction problems.

```
10 RANDOMIZE
20 W=0
30 PRINT "HI. TO STOP, INPUT
 9999 AS"
40 PRINT "YOUR ANSWER."
50 PRINT "WHAT IS YOUR GRADE
 LEVEL";
60 INPUT G1
70 G2=G1-.5*RND
80 R=INT(2+1.73*G2^4)
90 A=INT(100*G2*RND)
100 IF A>R THEN 90
110 B=R-A
115 PRINT
120 PRINT TAB(8-LEN(STR$(B))
);B
130 PRINT " + ";TAB(8-LEN(ST
R$(A)));A
140 PRINT " --------"
150 R=A+B
160 PRINT TAB(7-LEN(STR$(R))
);
170 INPUT G
200 IF G=9999 THEN 500
210 IF G=R THEN 310
220 W=W+1
230 IF W>1 THEN 260
240 PRINT "WRONG, TRY AGAIN"
250 GOTO 115
260 PRINT "YOU MISSED THAT O
NE TWICE."
270 PRINT "THE CORRECT ANSWE
R IS";R
280 W=0
290 IF G2>G1 THEN 400
300 GOTO 340
310 W=0
320 PRINT "CORRECT!"
330 IF G2<G1 THEN 400
340 G1=.9*G1+.1*SQR(SQR(R/2/
1.73))
400 PRINT
410 PRINT "HERE IS ANOTHER..
."
420 GOTO 70
500 PRINT
510 PRINT "OKAY. SO LONG FOR
 NOW."
520 G1=(INT(100*G1))/100
530 PRINT "YOUR NEW GRADE LE
VEL IS";G1
999 END
```

```
HI. TO STOP, INPUT 9999 AS
YOUR ANSWER.
WHAT IS YOUR GRADE LEVEL? 3

          5
 +       87
 ---------
    ? 92
CORRECT!

HERE IS ANOTHER...

         58
 +       67
 ---------
   ? 114
WRONG, TRY AGAIN

         58
 +       67
 ---------
   ? 125
CORRECT!

HERE IS ANOTHER...

          4
 +      104
 ---------
   ? 108
CORRECT!

HERE IS ANOTHER...

         28
 +       50
 ---------
    ? 9999

OKAY. SO LONG FOR NOW.
YOUR NEW GRADE LEVEL IS 3

** DONE **
```

# Time/Speed/Distance Problems

As well as being able to present simple numerical drill and practice problems, the computer can present word problems for solution as well.

In this program the formula relating time, speed and distance was applied to a problem involving both a car and train.

The problem can be stated as follows. A car traveling C miles per hour (computer generates an integer 40 through 65) can make a certain trip in D hours (computer generates an integer 5 through 20) less than a train traveling at T mph (computer generates an integer 20 through 39). How long does the trip take by car? When the two simultaneous equations are solved they produce the single equation for the answer shown in Line 110.

Notice the calculation in Line 120. This calculates the percent difference between the actual answer and the one entered by the user.

Notice also that the computer calculates the correct answer in Line110and prints it (on the screen) in Line 200. This answer may have many decimal places; as the program is written it is rounded off to two decimal places. If the 0.5 was not added in Line 110, the number would be truncated and not rounded off. This is an important calculation and one which you will find in many other programs throughout the book.

Consider other problems that can be used as the basis for this kind of drill and practice exercise. Teachers, for example, might wish to have students write drill and practice programs on their own. Different problems could be given to one or a small group of students to serve as the basis for a program.

After the programs are written, students can try out the programs of other class members. This approach ensures that each student not only understands the type of problem assigned to him, but also gets practice in solving other problem types as well. This is an effective technique for stimulating interest as well as for learning how to solve word problems.

```
5 RANDOMIZE
10 C=INT(RND*25)+40
20 D=INT(RND*15)+5
30 T=INT(RND*19)+20
40 PRINT "A CAR TRAVELING";C
;"MPH CAN"
50 PRINT "MAKE A CERTAIN TRI
P IN";D
60 PRINT "HOURS LESS THAN A
TRAIN"
70 PRINT "TRAVELING AT";T;"M
PH."
80 PRINT "HOW LONG DOES THE
TRIP TAKE"
90 PRINT "BY CAR";
100 INPUT A
110 V=INT(.5+100*(D*T/(C-T))
)/100
120 E=INT(ABS((V-A)*100/A)+.
5)
130 IF E>5 THEN 170
140 PRINT "GOOD. YOUR ANSWER
 WAS"
150 PRINT "WITHIN";E;"PERCEN
T."
160 GOTO 190
170 PRINT "SORRY, YOU WERE O
FF BY"
180 PRINT E;"PERCENT."
190 PRINT "THE CORRECT ANSWE
R IS"
200 PRINT V;"HOURS."
210 PRINT
220 PRINT "HERE IS ANOTHER..
"
230 GOTO 10
999 END
>
>
>RUN
A CAR TRAVELING 41 MPH CAN
MAKE A CERTAIN TRIP IN 6
HOURS LESS THAN A TRAIN
TRAVELING AT 33 MPH.
HOW LONG DOES THE TRIP TAKE
BY CAR? 25
GOOD. YOUR ANSWER WAS
WITHIN 1 PERCENT.
THE CORRECT ANSWER IS
 24.75 HOURS.

HERE IS ANOTHER...
A CAR TRAVELING 42 MPH CAN
MAKE A CERTAIN TRIP IN 7
HOURS LESS THAN A TRAIN
TRAVELING AT 24 MPH.
HOW LONG DOES THE TRIP TAKE
BY CAR? 9.3
GOOD. YOUR ANSWER WAS
WITHIN 0 PERCENT.
THE CORRECT ANSWER IS
 9.33 HOURS.
```

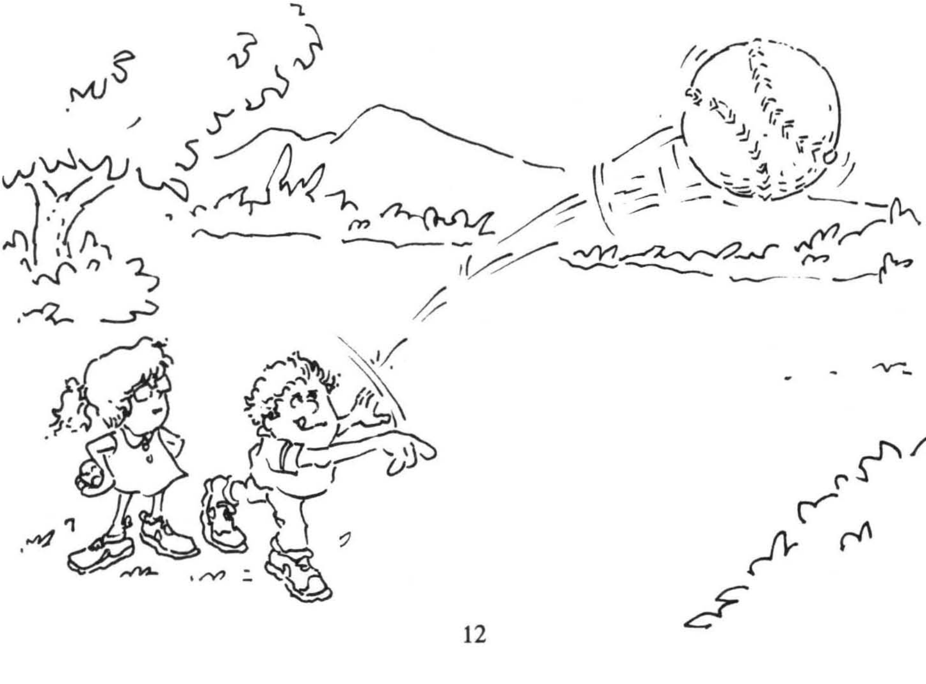
# Kinematics Problems

Kinematics relates to the dynamics of motion of bodies apart from considerations of mass and force. Like many other types of problems, these can be generated and presented by the computer to provide practice in solving them.

This program presents a simple kinematics problem for solution. The computer generates a new value for each problem. The problem is as follows. A ball (or any other object) is thrown up at velocity V meters per second (computer generates an integer between 5 and 40). The user must then calculate three factors about the resulting flight of the ball: maximum height, time until it returns, and velocity after T seconds (computer generates a time less than the total flight time).

The key benefit in using a computer to present problems of this type is motivation. The calculations required of the user are no different than those in the back of a chapter in a book or those on homework assignments. However, answering them when they are presented by the computer seems to make it more of a challenge and, frankly, more fun.

The computer program checks each of your responses to see if it is within 15% of the correct answer. If it is, your answer is considered correct. You may wish to change this percentage to require more or less accurate calculations. You may also wish to change the computer calculations to round off to one or two decimal places.



```
10  RANDOMIZE
20  Q=0
30  V=6+INT(35*RND)
40  PRINT "A BALL IS THROWN U
PWARDS"
50  PRINT "AT";V;"METERS PER
SECOND."
60  A=.05*V^2
70  PRINT "HOW HIGH WILL IT G
O (IN"
80  PRINT "METERS)";
90  GOSUB 500
100  A=V/5
110  PRINT "HOW LONG UNTIL IT
 RETURNS"
120  PRINT "(IN SECONDS)";
130  GOSUB 500
140  T=2+INT(2*V*RND)/10
145  IF T>A THEN 140
150  A=V-10*T
160  PRINT "WHAT WILL ITS VEL
OCITY BE"
170  PRINT "AFTER";T;"SECONDS
";
180  GOSUB 500
190  PRINT
200  PRINT Q;"RIGHT OUT OF 3.
"
210  IF Q<2 THEN 230
220  PRINT "      NOT BAD!"
230  STOP
500  INPUT G
510  IF ABS((G-A)/A)<.15 THEN
 540
520  PRINT "NOT EVEN CLOSE...
"
530  GOTO 560
540  PRINT "CLOSE ENOUGH."
550  Q=Q+1
560  PRINT "CORRECT ANSWER IS
";A
570  PRINT
580  RETURN
999  END
>

A BALL IS THROWN UPWARDS
AT 12 METERS PER SECOND.
HOW HIGH WILL IT GO (IN
METERS)? 8
CLOSE ENOUGH.
CORRECT ANSWER IS 7.2

HOW LONG UNTIL IT RETURNS
(IN SECONDS)? 2.5
CLOSE ENOUGH.
CORRECT ANSWER IS 2.4

WHAT WILL ITS VELOCITY BE
AFTER 2.4 SECONDS? -12
CLOSE ENOUGH.
CORRECT ANSWER IS-12

 3 RIGHT OUT OF 3.
     NOT BAD!
```

*"This nano-computer is great, but working the keyboard is a real problem!"*

# 2

# Problem Solving

In many courses in school, the textbooks present a great variety of devices for solving the problems that have been neatly grouped together at the end of each chapter. Typically these devices consist of formulae, equations, rules and theorems. After a careful study of these devices, teachers give exams which test your ability to recall them.

But what do you do if you are faced with the more realistic situation of not being told what device is likely to solve which problem or, worse yet, of having forgotten how to use a technique altogether? Is all hope lost? Of course not, although some people seem to believe that it is.

In this chapter, several of these nasty devices mentioned above are presented. For example, there are devices written into computer programs that will solve a quadratic or exponential equation and others that will calculate the roots and draw a plot of any function.

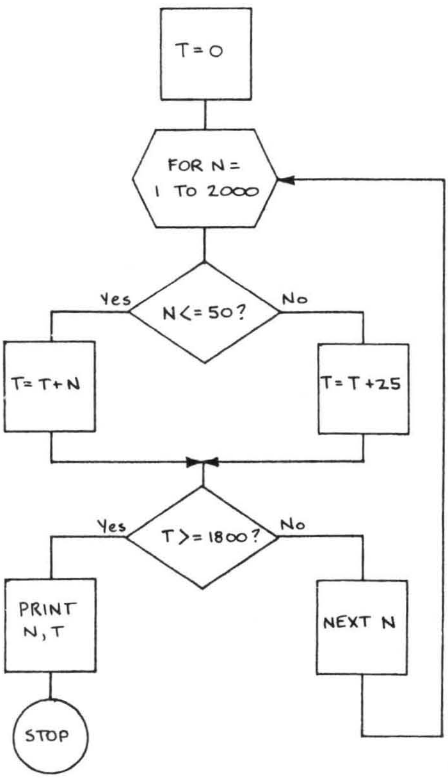
However, you must remember that while these devices are useful in solving certain problems, the really important thing is to understand the underlying logic and approach. Then when it comes time to solve real world problems you will be better prepared to face them. Incidentally, many of the methods and devices presented in this chapter are used in later chapters to solve other kinds of problems.

# How Many Tickets?

Here is a problem. At a school raffle to raise money, the organizers have as a prize an electronic game for which they paid $18.00. To add interest to the raffle, the organizers have decided to sell tickets for an amount (in cents) equal to the number on the ticket for tickets numbered 1 to 50. For ticket numbers over 50, the price is 25 cents each. The organizers want to know how many tickets they must sell to exactly break even.

It is probably easiest to visualize a problem of this sort with a flowchart. In the flowchart, T will equal the total money collected and will increase as more tickets are sold. The ticket number is N. When T equals or exceeds $18.00, N will be the answer.

Note that the flowchart has two logical branching points (IF statements in the program). The first compares the current ticket number to 50; if it is less, the ticket number is added to the total whereas if it is greater than 50, the total is increased by 25 cents.



The second branch point compares the total amount collected, T, to \$18.00 (actually 1800 cents). If T is equal to or greater than 1800, the break even point has been reached and the values of N and T are printed (on the screen).

A problem of this kind can be done by hand, however, because of the repetitive additions it is quite tedious. Also, doing it by hand frequently leads to an answer of 72 rather than the correct answer of 71. Try it yourself and see what you get.

```
10 T=0
20 FOR N=1 TO 2000
30 IF N<=50 THEN 60
40 T=T+25
50 GOTO 70
60 T=T+N
70 IF T=1800 THEN 90
80 NEXT N
90 PRINT N;"TICKETS SOLD TO"
100 PRINT "COLLECT $";T/100
999 END
>
>
>RUN
 71 TICKETS SOLD TO
COLLECT $ 18

** DONE **
```

Many problems can be solved quickly and correctly with a computer using logical analysis and a flowchart. More complex problems may have to be broken down into additional steps and require a longer flowchart, but the approach is fundamentally the same.

Here are two problems for you to solve.

The diameter of a long-playing record is 12 inches. The unused center has a diameter of 4 inches and there is a smooth outer edge 1/2 inch wide around the recording. If there are 91 grooves to the inch, how far does the needle move during the actual playing of the recording?

A movie theater charges \$2.50 for an adult admission and \$1.00 for a child. At closing, the cashier counted 385 ticket stubs and had \$626.50 in cash. How many children entered?

# Drinking and High Blood Pressure

This program illustrates how several simple equations can be put in a computer program to solve a more difficult overall problem.

Here is the problem. In a survey of 1000 adults, it was found that 35 had high blood pressure. Of those with high blood pressure, 80% drink 15 oz. or more of alcohol per week. Of those without high blood pressure, 60% drink a similar amount. What percent of drinkers and non-drinkers have high blood pressure?

This problem requires the solution of several simple equations. They could all be combined into one large equation, but it may be easier to understand the approach (and change variables later on) by writing a program with five separate equations.

If H equals the number of people with high blood pressure, then H = 35 (Line 10). Letting H1 equal the number of people with high blood pressure who drink leads to H1 = .8 x H (Line 20).

Letting L1 equal the number of people with low blood pressure who drink yields L1 = .6 x (1000 - H). Then, the total number of drinkers, D = H1 + L1.

Finally, the percentage of drinkers with high blood pressure is X = H1 x 100 / D.

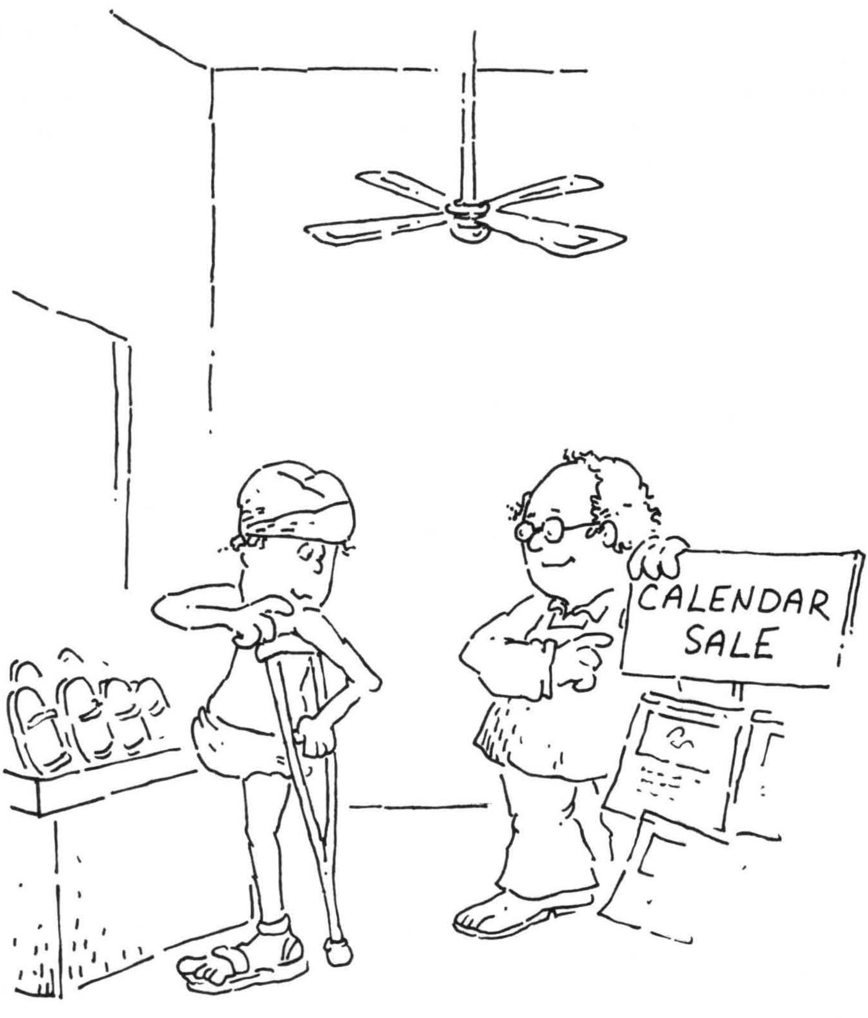
The program solves the problem in a jiffy. The solution for this type of problem can be easily written directly in Basic without any need for a flowchart or detailed analysis. Recognizing this type of problem readily will save a great deal of pencil pushing time.

```
10  H=35
20  H1=.8*H
30  L1=.6*(1000-H)
40  D=H1+L1
50  X=INT(10*(H*100/D))/10
60  B=H*100/1000
70  PRINT  TAB(8);"PEOPLE WITH
 HIGH"
80  PRINT  TAB(9);"BLOOD PRESS
URE"
90  PRINT  "DRINKERS       ";X;"
%"
100 PRINT  "ALL PEOPLE     ";B
999 END
>
>RUN
          PEOPLE WITH HIGH
           BLOOD PRESSURE
DRINKERS        5.7 %
ALL PEOPLE      3.5

**  DONE  **
```

Here is a problem that doesn't require a single equation but makes use of the approaches discussed so far in this chapter. Can you solve it with three Basic statements?

In early January, a shopkeeper marked down some calendars from $2.00 to a lower price. He sold his entire stock in one day for $603.77. How many did he have?

Here's another easy one. A town in India has a population of 20,000 people. Five percent of them are one-legged and half of the others go barefoot. How many sandals are worn in the town?



## Two Simultaneous Equations

So far in this chapter, only problems with linear equations have been considered. But the computer can be used to solve much more difficult equations. In fact, it is in problems involving second and third degree equations, exponentials, and the like where the computer really starts to pay off. Consider the following two simultaneous equations:

$$2^X = \frac{16y}{3} \qquad 3^X = 27y$$

It is not at all easy to solve these two equations by hand. But a simple Basic program can be written to solve the equations using trial and error. This is sometimes referred to as a brute force approach because every possible combination of numbers between an upper and lower limit is tried until a solution is reached or until the program runs out of values.

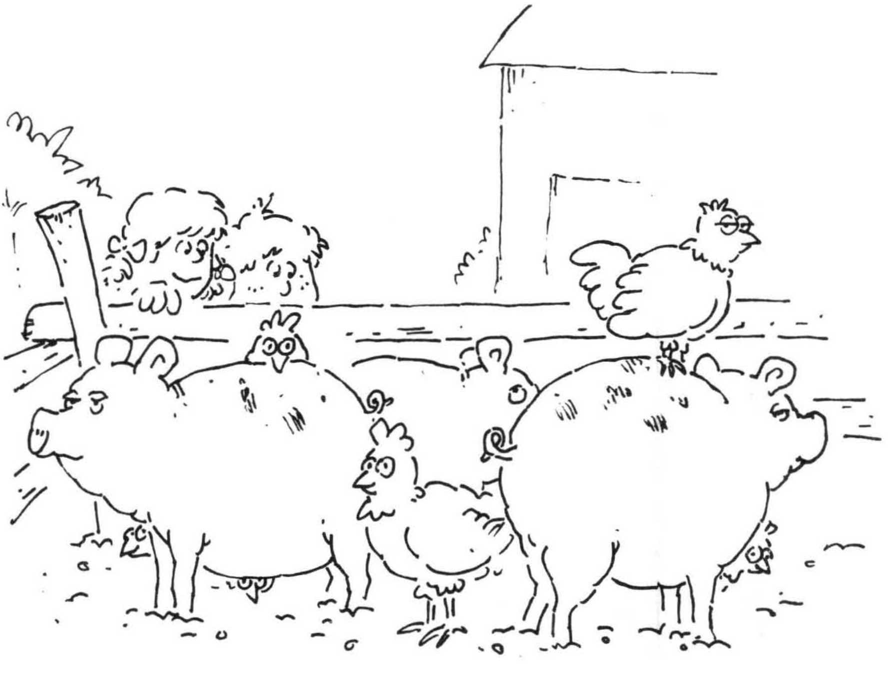
```
10 FOR X=1 TO 100
20 FOR Y=1 TO 100
30 IF 2^X<>16*Y/3 THEN 70
40 IF 3^X<>27*Y THEN 70
45 PRINT "FINALLY SOLVED IT.
"
50 PRINT "X=";X,"Y=";Y
60 STOP
70 NEXT Y
80 NEXT X
90 PRINT "NO INTEGER SOLUTIO
N"
100 PRINT "BETWEEN 1 AND 100
"
999 END
>
>
>RUN
FINALLY SOLVED IT...
X= 4          Y= 3

** DONE **
```

In this particular case, a solution is reached rather quickly with x = 4 and y = 3. However, if in the second equation the y coefficient is changed slightly from 27 to 28, the computer will try 10,000 possible solutions before finally concluding that no integer solution exists—at least within the range of 0 to 100. Warning: this will run for a *very* long time.

```
>40 IF 3^X<>28*Y THEN 70
>
>RUN
 NO INTEGER SOLUTION
 BETWEEN 1 AND 100.

 ** DONE **
```

Although the trial and error (brute force) approach is widely used, it is highly inefficient. In general, a systematic or guided trial and error approach is preferable to one that simply tries every possible solution. However, for some problems the simple "try every value" approach may be appropriate. (A comprehensive discussion of trial and error approaches can be found on pp 36-40 of *Computers in Mathematics: A Sourcebook of Ideas.)*

Three problem solving approaches have been discussed so far. Remembering them, how would you do this problem? A boy and his sister visited a farm where they saw a pen filled with pigs and chickens. When they returned home, the boy observed that there were 18 animals in all, and his sister reported that she had counted a total of 50 legs. How many pigs were there in the pen?



# Quadratic Equation Solver

For any values A, B, and C of a first degree quadratic equation ($Ax^2 + Bx + C = 0$), this program will compute the roots of the equation. The solution is based on the quadratic theorem which solves for roots with the following formula:

$$X = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

Assuming that A, B, and C are real numbers, the following principles apply:

1. If $B^2 - 4AC$ is positive, then the roots are real and unequal.
2. If $B^2 - 4AC$ equals 0, then the roots are real and equal.
3. If $B^2 - 4AC$ is negative, then the roots are imaginary and unequal.

The program takes into account all these possibilities and correctly identifies the type of roots along with their values for any set of coefficients.

Is this program useful by itself? Except for solving quadratic equations for algebra class, probably not. However, as a routine in a larger program to solve quadratic equations that might be encountered, it could be very useful.

```
10 PRINT "QUADRATIC EQUATION
 SOLVER"
20 PRINT "PLEASE ENTER THE F
OLLOWING:"
30 INPUT "VALUE OF A=":A
40 INPUT "VALUE OF B=":B
50 INPUT "VALUE OF C=":C
60 PRINT
70 R=B^2-4*A*C
80 IF A<>0 THEN 120
90 PRINT "THAT IS A FIRST"
100 PRINT "DEGREE EQUATION"
110 STOP
120 PRINT "THE ROOTS ARE ";
130 IF R<0 THEN 205
140 PRINT
150 IF R=0 THEN 190
160 PRINT (-B+SQR(R))/2*A
170 PRINT (-B-SQR(R))/2*A
180 STOP
190 PRINT -B/2*A
200 STOP
205 PRINT "IMAGINARY"
210 PRINT (-B/2*A);"+";SQR(-
R)/2*A;"*I"
220 PRINT (-B/2*A);"-";SQR(-
R)/2*A;"*I"
999 END
```

```
>RUN
 QUADRATIC EQUATION SOLVER
 PLEASE ENTER THE FOLLOWING:
 VALUE OF A=1
 VALUE OF B=2
 VALUE OF C=1

 THE ROOTS ARE
 -1

 ** DONE **

>RUN
 QUADRATIC EQUATION SOLVER
 PLEASE ENTER THE FOLLOWING:
 VALUE OF A=0
 VALUE OF B=2
 VALUE OF C=4

 THAT IS A FIRST
 DEGREE EQUATION

 ** DONE **

>RUN
 QUADRATIC EQUATION SOLVER
 PLEASE ENTER THE FOLLOWING:
 VALUE OF A=2
 VALUE OF B=5
 VALUE OF C=2

 THE ROOTS ARE
 -2
 -8

 ** DONE **

>RUN
 QUADRATIC EQUATION SOLVER
 PLEASE ENTER THE FOLLOWING:
 VALUE OF A=4
 VALUE OF B=2
 VALUE OF C=4

 THE ROOTS ARE IMAGINARY
 -4 + 15.4919338 *I
 -4 - 15.4919338 *I

 ** DONE **
```

# Exponential Equation Solver

Another general routine for solving a particular type of equation is this one to solve for an exponent in an exponential equation.

Given the values A, B, m, and n, this program will solve for x in any exponential equation of the form:

$$A^{mx+n} = B$$

For example, the program will solve any of the following problems:

1. $5^X = 40$
2. $5^{3X+1} = 7.6$
3. $17^{X-3} = 8.12$
4. $11^{1-2X} = 247$

If you were to solve an exponential equation by hand, you would probably go through the following steps:

$$5^X = 40$$
$$\log 5^X = \log 40$$
$$x \log 5 = \log 40$$
$$x = \log 40/\log 5$$
$$x = 1.6021/.6990 = 2.292$$

However, in more generalized form, the solution for x is:

$$X = \frac{\frac{\log B}{\log A}}{M-(N/M)}$$

Note that the program to solve this problem is actually divided into two sub-programs. The first is a data loader program. It requires that data be entered in the following order: A, B, m, and n for each equation to be solved. If a coefficient is not present, it must be entered as a zero.

The data for the four problems listed above were entered into the two data statements (50 and 60).

The program as it is presented here can be improved in several ways. First, it always solves the same four equations. How can you generalize it to solve for other equations? Second, if you were to make use of this routine in another program, you would probably not be able to use a READ statement; how could you get rid of it?.

```
10 PRINT "EXPONENTIAL EQUATI
ON SOLVER"
15 PRINT "A    B    M    N
    X"
20 READ A,B,M,N
30 X=(LOG(B)/LOG(A))/M-(N/M)
35 X=INT(.5+100*X)/100
40 PRINT A;TAB(5);B;TAB(10);
M;TAB(15);N;TAB(20);X
45 GOTO 20
50 DATA 5,40,1,0,5,7.6,3,1
60 DATA 17,8.12,1,-3,11,247,
-2,1
99 END
>
>
>RUN
EXPONENTIAL EQUATION SOLVER
A    B    M    N    X
 5   40   1    0    2.29
 5   7.6  3    1    .09
 17  8.12
          1    -3   3.74
 11  247  -2   1    -.65

* DATA ERROR IN 20
```

# Roots of Any Function

This program will find the roots of a function, any function! The function may be linear, quadratic, cubic, trigonometric or any combination as long at it can be represented in the Basic language. The program as it appears here finds the roots between -20 and 20 although you can change these boundaries in Statement 120.

The method used involves evaluating the function at small incremental intervals, finding places where the value of the function changes sign and then, by successive approximations, finding the zero point. This approach borrows from Newton's method in the final narrowing down but, unlike Newton's method, will not fail to converge in the event one makes an unlucky first guess.

Before running the program you must first type in your function in Statement 100. For example,

```
DEF FNA (X) = 2*X↑3+11*X↑2-31*X-180
DEF FNA (X) = X-4
DEF FNA (X) = SIN(X)-.5
```

You may have to refer to the Basic manual with your system to see exactly how a function should be stated.

The routine used in this program is very powerful and could possibly be used as a subroutine in many other programs.

How can you use this program to help you solve this problem for x?

$$x = \sqrt{12 + \sqrt{12 + \sqrt{12 + \sqrt{12 + \sqrt{12 + \ldots}}}}}$$

```
 100  DEF  FNA(X)=2*X^3+11*X^2-
 31*X-180
 110  Z1=-200
 120  FOR  I=-19.9  TO  20
 130  IF  SGN(FNA(I))=SGN(FNA(I
 +1))THEN  300
 140  K=I
 150  J=I+1
 160  IF  FNA(K)<FNA(J)THEN  200
 170  Z=K
 180  K=J
 190  J=Z
 200  Z=(K+J)/2
 210  IF  FNA(Z)<0  THEN  240
 220  J=Z
 230  GOTO  250
 240  K=Z
 250  IF  ABS(FNA(Z))>.000005  T
 HEN  200
 260  Z=SGN(Z)*INT(ABS(Z)*1000
 0+.5)/10000
 270  IF  Z=Z1  THEN  300
 280  PRINT  "F(";Z;")=0"
 290  Z1=Z
 300  NEXT  I
 999  END
>
>RUN
 F(-5  )=0
 F(-4.5  )=0
 F(  4  )=0

 **  DONE  **

>100  DEF  FNA(X)=X-4
>RUN
 F(  4  )=0

 **  DONE  **

>100  DEF  FNA(X)=SIN(X)-.5
>RUN
 F(-18.326  )=0
 F(-16.2316  )=0
 F(-12.0428  )=0
 F(-9.9484  )=0
 F(-5.7596  )=0
 F(-3.6652  )=0
 F(  .5236  )=0
 F(  2.618  )=0
 F(  6.8068  )=0
 F(  8.9012  )=0
 F(  13.09  )=0
 F(  15.1844  )=0
 F(  19.3732  )=0

 **  DONE  **
```

# Plot Any Function

Here is a nifty program that will produce a plot of any function on the screen or printer.

Before running this program, you must type in your function in line 200. Like the previous program, which finds the roots of any function, this one will plot any function. You must tell the program between what values you want the function plotted, i.e., a minimum and maximum value of the x coordinate. You also input the x plotting increment you wish.

As it appears here, the program does not allow the user to select the y coordinates; the program plots y values between -30 and 30.

Here are several functions you might want to try plotting:

| *Function* | *X Limits* | | *Increment* |
|---|---|---|---|
| DEF FNA(X) = 2*X | -15 | 15 | 1 |
| DEF FNA(X) = 30*SIN(X) | -5 | 5 | .25 |
| DEF FNA(X) = X-X*X | -5 | 6 | .5 |
| DEF FNA(X) = 30*EXP(-X*X/100) | -30 | 30 | 1.5 |
| DEF FNA(X) = X*X-X | -5 | 6 | 1 |
| DEF FNA(X) = X↑2-X-15 | | | |

The last function listed is the one plotted in the sample run with the program. Exponential functions are a great deal of fun and sometimes lead to unexpected and interesting results, particularly when combined with trigonometric functions. Experiment! Have fun!

```
200 DEF FNA(X)=X^2-X-15
210 PRINT
220 INPUT "INITIAL VALUE OF
X=":X1
230 INPUT "FINAL VALUE OF X=
":X2
240 INPUT "INCREMENTAL STEP
=":S
250 PRINT
290 PRINT TAB(12);"X=";X1
300 FOR X=X1 TO X2 STEP -S
310 IF ABS(X)<.000001 THEN 3
60
320 Y=FNA(X)/2+14
322 IF Y<28 THEN 330
325 Y=27
330 IF Y>14 THEN 410
340 PRINT TAB(Y);"*";TAB(14)
;"|"
350 GOTO 420
360 PRINT "-Y --------------
---------- +Y"
370 PRINT "-30 -20 -10  0 +1
0 +20 +30"
380 X=X-S
390 GOTO 420
410 PRINT TAB(14);"|";TAB(Y)
;"*"
420 NEXT X
430 PRINT TAB(12);"X=";X2
999 END
>
>
```

```
INITIAL VALUE OF X=7
FINAL VALUE OF X=-6
INCREMENTAL STEP =1

           X= 7
             |             *
             |       *
             |  *
            *|
         *   |
       *     |
      *      |
-Y -------------------------- +Y
-30 -20 -10  0 +10 +20 +30
         *   |
            *|
             |  *
             |       *
             |             *
           X=-6

** DONE **
```

*"I'm in the kitchen, dear—using the computer."*

# 3

# Sets and Repetitive Trials

For solving relatively simple problems, the computer may not be any help at all. In fact, it may take more time to write a program to solve a problem than it would to solve it by hand or with a calculator. This chapter should help you recognize problems that are suitable for computer solution and those that are not.

In some problems, you may think that the computer will not be any help. However, one thing writing a computer program will always do is force you to reason out the approach to solving the problem logically and precisely. The computer can't solve problems unless it is told exactly how to proceed; hence you must understand a problem completely before you can program it for the computer.

Several of the sections in this chapter discuss sets of data. While the sets used in the examples have relatively few elements or values, you should bear in mind that real world problems often have thousands or millions of pieces of data and the only practical way to solve problems of this size is with a computer. For example, consider how you would most efficiently schedule the shipments on a railroad train leaving Boston with 4000 diverse cargoes bound for Phoenix and 780 points in between. Now consider that there are 200 freight trains per day leaving Boston. Now add to that the 10,000 other trains leaving other cities every day that must use the same network of track and you can see that dealing with real world sets of data is no easy task.

# Group of Girls and Boys

In the previous chapter we said that there may be some problems for which brute force trial and error is appropriate. This would be the case if the problems were relatively small and trying every possible solution would not tie up a great deal of valuable computer time. Here is a problem involving two linear equations that lends itself to a trial and error approach.

The problem is as follows: When 15 girls leave a group of boys and girls, there are two boys for every girl (lucky girls). Next, 45 boys decide to leave; then there are 5 girls for every boy (lucky boys!). How many girls were there in the group before anyone left?

Before rushing to the computer, you must recognize that this problem requires the solution of two simultaneous equations. If G equals the original number of girls and B the original number of boys, then the two equations are:

$$(G - 15) \times 2 = B$$

$$(B - 45) \times 5 = (G - 15)$$

```
10 I=0
20 FOR G=1 TO 100
30 FOR B=1 TO 100
40 I=I+1
50 IF 2*(G-15)<>B THEN 90
60 IF 5*(B-45)<>(G-15)THEN 9
0
70 PRINT G;"GIRLS, ";B;"BOYS
"
80 GOTO 120
90 NEXT B
100 NEXT G
110 PRINT "NO SOLUTION BETWE
EN 1 & 100"
120 PRINT I;"COMBINATIONS TR
IED"
999 END

>RUN
 40 GIRLS, 50 BOYS
 3950 COMBINATIONS TRIED

** DONE **
```

The computer program uses two FOR loops (Statements 20-100 and 30-90) to try every combination of values for B and G between 1 and 100 until a solution is found or until the program runs out of values. The variable I (Statement 40) is a counter which records the number of trials required to reach a solution.

The program is straightforward and finds a solution after 3950 trials. However, it would have been a simple matter to substitute the value of B from the first equation in the second one and quickly solve the problem by hand or with the aid of a calculator. It is important to recognize that if a problem can easily be solved by other methods, the computer offers little or no advantage.

Try this problem. You may or may not want to use your computer. If Matthew can beat Jeff by one-tenth of a mile in a two-mile race and Jeff can beat Steven by one-fifth of a mile in a two-mile race, by what distance could Matthew beat Steven in a two-mile race? (Hint: the answer is not 3/10 mile.)

# Brown's Books

The use of a trial and error approach can generally be improved significantly if the combinations to be tried can be narrowed down in some way. The solution to this problem illustrates how the speed of obtaining a solution can be improved well over 100 fold by combining equations and eliminating certain solution possibilities.

Here is the problem. Brown sold 48 books at a flea market, some for \$3 each, some for \$5 each and others for \$8. He collected a total of \$175. He remembered having an even number of \$5 books. Can you determine how many of each kind of book he had?

The equations for solution are (letting T equal the number of \$3 books, F the number of \$5 books, and E the number of \$8 books):

$$T + F + E = 48$$

$$3*T + 5*F + 8*E = 175$$

The first program was written simply to try all possible combinations of T, F, and E from 1 to 48. It yields three solutions for the problem, although the two solutions with an odd number of \$5 books can be eliminated leaving just the one desired solution.

```
10 PRINT "BOOKS PROBLEM"
20 PRINT "ALL SOLUTIONS"
30 PRINT
40 PRINT "$10";TAB(10);"$5";
TAB(20);"$8"
50 PRINT
60 FOR T=1 TO 48
70 FOR F=1 TO 48
80 FOR E=1 TO 48
90 IF (T+F+E)<>48 THEN 120
100 IF (T*3+F*5+E*8)<>175 TH
EN 120
110 PRINT T;TAB(10);F;TAB(20
);E
120 NEXT E
130 NEXT F
140 NEXT T
999 END
>
>
>RUN
BOOKS PROBLEM
ALL SOLUTIONS

$10          $5          $8

 40           3           5
 37           8           3
 34           13          1

** DONE **
```

This program took approximately 22 minutes and 13 seconds to run on the TI 99/2 computer. A typical minicomputer (PDP-8/e) could run this problem in about 7.3 seconds. In either case, this is a long time to tie up the computer.

It is rather easy to combine the two equations into one by solving for T. The single equation is then:

$$2*F + 5*E = 31$$

In this equation, the limits can be reduced (from 48 used in the first run) since F cannot possibly be greater than 31/2 or 15.5 and E cannot be greater than 31/5 or 6.2. Making the appropriate program modifications leads to the second program.

```
>60
>70 FOR F=1 TO 15
>80 FOR E=1 TO 6
>90 IF (F*2+E*5)<>31 THEN 120
>100
>105 T=48-F-E
>140
>
>RUN
 BOOKS PROBLEM
 ALL SOLUTIONS

 $10           $5            $8

  40            3             5
  37            8             3
  34           13             1

 ** DONE **
```

Using this program produces a dramatic improvement in the time to solution. On the TI 99/2, the time is approximately 2.5 seconds and on the PDP-8, about 0.16 seconds.

Since the problem states that F must be even, a final modification which steps F by two in Statement 20, can be made. This version of the program takes only 1.25 seconds to run on the Timex and 0.06 seconds to run on the PDP-8.

```
>20 PRINT "SOLUTIONS, EVEN VA
LUE OF F"
>70 FOR F=2 TO 14 STEP 2
```

Notice the enormous improvement in computing time required for a solution, over 1000 fold on the TI 99/2 and 100 fold on a PDP-8. Brute force certainly is inefficient! It is generally worthwhile to think through most problems, particularly big ones, before rushing to the computer. The computer may be fast, but we just improved its performance by 1000 times by using a little common sense.

# Intersection of Sets

Two sets of numbers can be combined to yield a third set by the operation of intersection. The intersection of two sets A and B is the set that contains all elements that belong to both A and B. It does not contain any other elements. The intersection is usually written A ∩ B.

For example if M = < 0,2,4,6 > and K = < 1,2,3,4 >, then M ∩ K = < 2,4 >.

This program finds the intersection of two sets of numbers. It has been written to find the intersection of the two repetitive sets described in Statements 30 and 40. In the sample run, Statement 30 describes the set

x = < 1,3,5, . . . 19 > and Statement 40 describes the set

y = < 2,5,8, . . . 29 >.

Notice that successive values of x increase by 2 and y by 3.

```
10 PRINT "THE INTERSECTION O
F SETS"
20 PRINT "X AND Y IS:"
30 FOR X=1 TO 19 STEP 2
40 FOR Y=2 TO 29 STEP 3
50 IF X=Y THEN 100
60 NEXT Y
70 NEXT X
80 STOP
100 PRINT X
110 GOTO 70
999 END
>
>RUN
THE INTERSECTION OF SETS
X AND Y IS:
 5
 11
 17

** DONE **
```

However, if the set cannot be so neatly described, it may be desirable to rewrite the program to examine any set of data. This is done with the READ statement which reads into x the data points in the DATA statement.The program is set up to use the same y set as the first program, but the x set is defined in the data statement.

You should be able to see from the first combination of sets that if there is a numerical pattern in the sets which intersect, then there is also a pattern in the resulting intersecting set. In the example, the x values increase by 2 and the y values by 3, hence the values in the intersecting set increase by 2 x 3 = 6. Although the intersection of these sets could easily be calculated by hand, the computer can be an aid in evaluating more complicated sets.

```
10 PRINT "THE INTERSECTION O
F SETS"
20 PRINT "X AND Y IS:"
30 READ X
40 FOR Y=2 TO 29 STEP 3
50 IF X=Y THEN 100
60 NEXT Y
70 GOTO 30
100 PRINT X
110 GOTO 30
200 DATA 2,3,8,9,14,15,20,21
,26,27
999 END
>
>
>RUN
THE INTERSECTION OF SETS
X AND Y IS:
 2
 8
 14
 20
 26

* DATA ERROR IN 30
```

# Prime Factors

A prime factor is a positive integer that has no factor except itself and one. The first ten prime factors (or numbers) are 2, 3, 5, 7, 11, 13, 17, 19, 23, and 29. The definition gives the basic method for determining whether a number is prime: divide by all smaller integers down to 2, testing whether the remainder is zero for at least one of them. If not, the number is prime.

But this is highly inefficient. It is obvious that a number is not prime if it is any even number greater than 2; hence only odd divisors need to be tried. Also, it is not necessary to try divisors greater than the square root of the number.

Since the division method is inefficient, various schemes have been devised to avoid division. The basic idea underlying all such schemes is called the sieve of Eratosthenes (276 B.C.-195 B.C.). Imagine a list of odd numbers from 3 up. Strike out every third number after 3, every fifth number after 5, and so on. This will leave only prime numbers.

```
10 DIM A(140)
20 C=0
30 PRINT "PROGRAM COMPUTES PRIME"
40 PRINT "FACTORS OF ANY INTEGER"
50 PRINT
60 PRINT "ENTER 0 (ZERO) TO STOP"
90 PRINT
100 INPUT "YOUR NUMBER =";M
110 X=0
120 M=INT(M)
130 IF M>0 THEN 160
135 IF M=0 THEN 330
140 PRINT "MUST BE POSITIVE"
150 GOTO 100
160 PRINT M;
170 I=1
180 I=I+1
190 IF I>M THEN 250
200 IF M/I<>INT(M/I)THEN 180
210 X=X+1
220 A(X)=I
230 M=M/I
240 GOTO 200
250 IF X=1 THEN 300
260 FOR L=1 TO X
270 PRINT A(L);
280 NEXT L
290 GOTO 310
300 PRINT "IS PRIME"
310 PRINT
320 GOTO 90
330 STOP
```

```
PROGRAM COMPUTES PRIME
FACTORS OF ANY INTEGER

ENTER 0 (ZERO) TO STOP

YOUR NUMBER =105
 105   3   5   7

YOUR NUMBER =72
 72   2   2   2   3   3

YOUR NUMBER =89
 89  IS PRIME

YOUR NUMBER =47
 47  IS PRIME

YOUR NUMBER =0

** DONE **
```

The program here finds the prime factors of any integer, or prints out "N is prime" if the integer has no proper divisors.

Run this program for a large number of different integers and see if you can discover relationships between numbers and their prime factors. You should also try to figure out the method employed in the program to find the prime factors of any integer. To do this, you might want to draw a flowchart to show what is happening in the program. This will help you see the method used to find a prime factor and might help you in writing a program to generate primes.

In writing a program to generate prime factors, you can use the sieve method. However, as the numbers become very large, you will have to figure out a way to represent integers with more digits than your computer can handle at one time. (One approach is described on pp. 19-21 of *Computers in Mathematics.*)

Goldbach was a mathematician who made a conjecture that every even number greater than 4 can be written as the sum of two prime numbers (16 = 11 + 5, 30 = 17 + 13, etc.). No one has ever proved it but no one has disproved it either. That is why it is called a conjecture. Can you write a program that will prove or disprove this conjecture? Or how about writing a program to prove Goldbach's conjecture for even numbers up to 50? You should be able to write this program with 12 or fewer statements.

Here is another problem involving prime numbers. Assume a life span of 80 years. In what year of the 20th century (1900-1999) would a person have to be born to have the maximum number of birthdays occurring in prime years? The minimum number?

# Greatest Common Divisor

The greatest common divisor of a set of numbers is, as its name implies, the greatest integer that will divide into a set of two or more numbers. For example, the set of numbers 12, 20, and 28 have a greatest common divisor of 4. Nothing larger than 4 will divide evenly into all three numbers.

This program will find the greatest common divisor for any set of integers. To run it, you simply input the number of integers in your set, type them in when requested and let the program calculate the GCD. The heart of the calculation is in Statement 180.

Do you know the meaning of a relatively prime set of numbers? Can you figure out the meaning from the third sample run of the program or from runs of your own? How is a set of relatively prime numbers different from a set of prime factors? Can you find a set of 10 integers that is relatively prime?

```
10 PRINT "PROGRAM COMPUTES G
REATEST"
20 PRINT "COMMON DIVISOR."
30 PRINT
40 INPUT "NUMBERS IN SET =";
N
50 N=INT(N)
60 DIM X(100)
70 PRINT "ENTER NUMBERS"
80 S=1E25
90 FOR K=1 TO N
100 INPUT X(K)
110 IF X(K)>S THEN 130
120 S=X(K)
130 NEXT K
140 PRINT
150 G=0
160 FOR M=2 TO S
170 FOR I=1 TO N
180 IF X(I)/M<>INT(X(I)/M)TH
EN 210
190 NEXT I
200 G=M
210 NEXT M
220 PRINT "NUMBERS ";
230 IF G>0 THEN 260
240 PRINT "ARE RELATIVELY PR
IME"
250 GOTO 270
260 PRINT "G.C.D. =";G
270 PRINT
280 PRINT "ANOTHER SET (Y OR
 N)";
290 INPUT A$
300 IF A$="Y" THEN 30
310 STOP
```

```
>RUN
PROGRAM COMPUTES GREATEST
COMMON DIVISOR.

NUMBERS IN SET =3
ENTER NUMBERS
? 12
? 36
? 96

NUMBERS G.C.D. = 12

ANOTHER SET (Y OR N)? Y

NUMBERS IN SET =3
ENTER NUMBERS
? 20
? 36
? 97

NUMBERS ARE RELATIVELY PRIME

ANOTHER SET (Y OR N)? ■
```

In the last section we discussed prime numbers. Here is an interesting challenge for you involving prime numbers. Until late 1982, the longest progression of prime numbers in which all differed by the same number was 17. Prof. Paul Pritchard in the computer science department at Cornell University wrote a program to determine if there was a longer progression. Using a DEC VAX-11/780, he found the string of 18 numbers shown below. He also discovered fourteen other 17-number progressions and ten 18-number progressions, but none yet with 19 numbers. He believes there is at least one; can you find it?

| | |
|---|---|
| 107928278317 | 197233324147 |
| 117851061187 | 207156107017 |
| 127773844057 | 217078889887 |
| 137696626927 | 227001672757 |
| 147619409797 | 236924455627 |
| 157542192667 | 246847238497 |
| 167464975537 | 256770021367 |
| 177387758407 | 266692804237 |
| 187310541277 | 276615587107 |

# Cryptarithmetic Problems

Cryptarithmetic or alphametic problems are arithmetic expressions in which the digits are replaced by letters of the alphabet. Each digit is associated with a letter to produce an interesting statement, for example:

$$\begin{array}{r} \text{SEND} \\ +\ \text{MORE} \\ \hline \text{MONEY} \end{array}$$

If the college student who sent this message to his father needed $106.52 for plane fare home, this was the right message to send since this combination of letters has one unique solution, in particular:

$$\begin{array}{r} 9567 \\ +\ 1085 \\ \hline 10652 \end{array}$$

However, if the student let things go to the last moment and was in more of a rush, he might have reworded the message:

$$\begin{array}{r} \text{WIRE} \\ +\ \text{MORE} \\ \hline \text{MONEY} \end{array}$$

In this case, how much should his dad send? Earlier in the book, trial and error approaches to solving problems were discussed. It was noted that the brute force approach of trying every alternative was sometimes appropriate. Is it in this case?

No! The number of possible alternative solutions is the factorial of the number of different letters in the alphametic expression, i.e., 8! or 40,320. A program to try out every one of these possibilities would run for a l-o-n-g time.

In this case it is much more efficient to apply some common sense to narrow down the number of alternatives. The best approach to this process is to divide up the search space into large classes (or sets), according to a common property shared by members of each class, and then attempt to eliminate entire classes by the method of contradiction.

Consider the "WIRE + MORE = MONEY" problem. Can E = 0? Since E + E = Y, Y must also equal 0, contradicting the fact that Y and E must be different digits. Thus, the entire class of solutions in which E = 0 can be ruled out.

Consider E = 3. Now Y = 6 and there is no carry to the next column. So in this column R + R = E or E + 10 if a carry is involved. But in either case E must be an even number since 2R is always even; this contradicts the assumption that E = 3.

By following this type of classificatory contradiction process for each of the

digits in the order E, R, I, O and N, the computer program will search out all possible solutions to the problem. Unlike the "send more money" problem, the "wire more money" problem has five possible solutions. A smart father would choose the lowest solution and wire his son $103.48.

```
10 PRINT "PROGRAM FINDS ALL SOLUTIONS"
20 PRINT "TO THE FOLLOWING PROBLEM:"
30 PRINT
40 PRINT "    W  I  R  E"
50 PRINT " +  M  O  R  E"
60 PRINT "--------------"
70 PRINT " M  O  N  E  Y"
80 PRINT
90 M=1
100 FOR E=2 TO 9
110 Y=E+E
120 IF Y>10 THEN 150
130 C1=0
140 GOTO 170
150 C1=1
160 Y=Y-10
170 FOR R=1 TO 9
180 IF R=M THEN 470
182 IF R=E THEN 470
184 IF R=Y THEN 470
190 IF R+R+C1=E THEN 220
200 IF R+R+C1=E+10 THEN 240
210 GOTO 470
220 C2=0
230 GOTO 250
240 C2=1
250 FOR I=0 TO 9
260 IF I=M THEN 460
262 IF I=E THEN 460
264 IF I=Y THEN 460
266 IF I=R THEN 460
270 FOR O=0 TO 9
280 IF O=M THEN 450
282 IF O=E THEN 450
284 IF O=Y THEN 450
286 IF O=R THEN 450
288 IF O=I THEN 450
290 N=I+O+C2
300 IF N>=10 THEN 330
310 C3=0
320 GOTO 350
330 C3=1
340 N=N-10
350 IF N=M THEN 450
360 FOR W=0 TO 9
362 IF W=M THEN 440
364 IF W=E THEN 440
366 IF W=Y THEN 440
368 IF W=R THEN 440
370 IF W=I THEN 440
372 IF W=O THEN 440
374 IF W=N THEN 440
380 IF O+10<>W+M+C3 THEN 440
```

```
390 PRINT "    ";W;I;R;E
400 PRINT "   ";M;O;R;E
410 PRINT "-----------------"
420 PRINT M;O;N;E;Y
430 PRINT
440 NEXT W
450 NEXT O
460 NEXT I
470 NEXT R
480 NEXT E
490 STOP
>
```

```
PROGRAM FINDS ALL SOLUTIONS
TO THE FOLLOWING PROBLEM:

     W   I   R   E
   + M   O   R   E
 -----------------
  M  O   N   E   Y

     9   7   6   2
     1   0   6   2
 -----------------
  1  0   8   2   4

     9   2   7   4
     1   0   7   4
 -----------------
  1  0   3   4   8

     9   5   7   4
     1   0   7   4
 -----------------
  1  0   6   4   8

     9   2   8   7
     1   0   8   7
 -----------------
  1  0   3   7   4

     9   5   8   7
     1   0   8   7
 -----------------
  1  0   6   7   4

** DONE **
```

There are other approaches to solving cryptarithmetic problems, but all of them benefit greatly from reducing the search space as much as possible before putting the problem on the computer. See if you can devise another successful approach and write a program to implement it.

Here are some problems for you to try.

```
  DONALD
+ GERALD
--------
  ROBERT
```

```
   TWO
 x TWO
 -----
 THREE
```

```
 ABCDE
   x 4
 -----
 EDCBA
```

```
  ABC
 x DE
 ----
  FEC
 DEC
 ----
 HGBC
```

```
      THE
    EARTH
    VENUS
   SATURN
 + URANUS
 --------
  NEPTUNE
```

```
   SPRING
    RAINS
    BRING
 +  GREEN
 --------
   PLAINS
```

VIOLIN + VIOLIN + VIOLA + CELLO = QUARTET

THREE + NINE = EIGHT + FOUR

```
   ONE
   TWO
 + FIVE
 ------
 EIGHT
```

```
  FORTY
    TEN
  + TEN
  -----
  SIXTY
```

```
  FIVE
 -FOUR
 -----
   ONE
 + ONE
 -----
   TWO
```

# Sailors and Monkey Problem

There are many variations of the sailors and monkey problem. Here is one of them.

Five sailors and a monkey were on an island. One evening the sailors rounded up all the coconuts they could find and put them in a large pile. Being exhausted from working so hard, they decided to wait and divide them up equally in the morning. During the night, a sailor awoke and separated the nuts into five equal piles, but had one nut left over which he gave to the monkey. He took one pile, hid it, and pushed the other four together and went back to sleep. He was followed in this action by the other four sailors, each of whom did exactly the same thing. Next morning the remaining nuts were divided equally with one remaining nut going to the monkey. What is the smallest number of coconuts with which they could have begun?

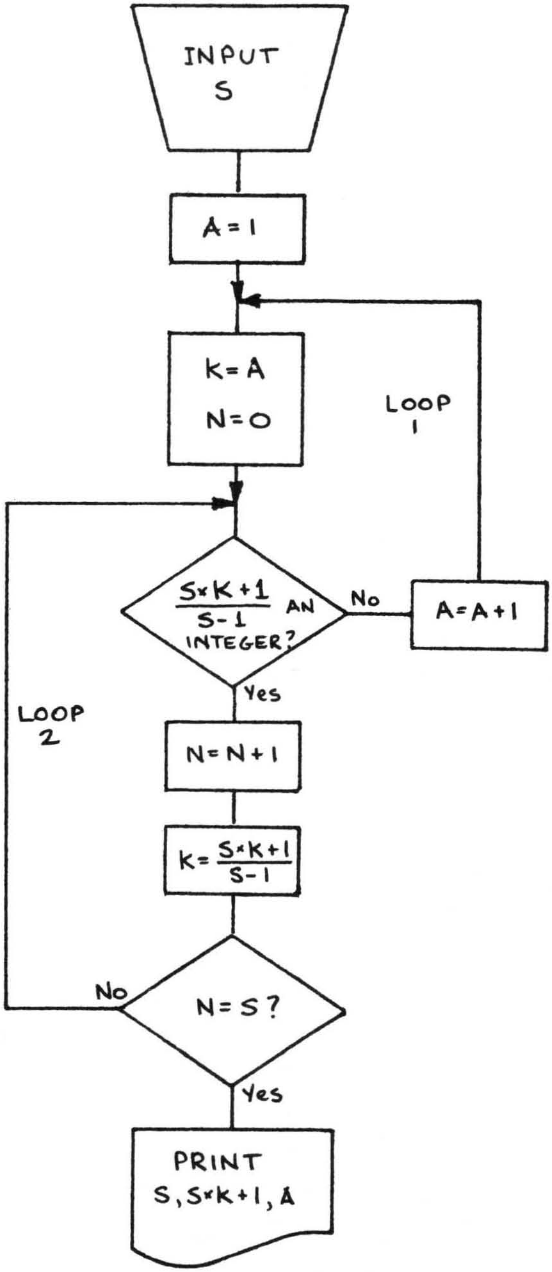
Although there is an elegant algebraic solution to this problem, a more suitable approach for the computer is that of working backwards. A typical solution to a problem can be thought of as a path that leads from the given information to the goal. However, in this case the goal, or final state, is known, thus it is easier to start there and work backwards to the initial state.

As mentioned at the outset, many sailor and monkey problems exist, in fact, an infinite number of them. For example, instead of five sailors, there could be three or six or 14. Thus it is desirable to devise a general solution instead of just one to solve one specific problem.

In the flowchart and computer program, S is the number of sailors and A is the number of coconuts that each sailor received in the final division of the pile. Since one coconut was given to the monkey at each division, the total number of coconuts left in the morning must be S x A + 1. But this pile came from pushing together S - 1 equal piles. Thus, the key condition that must hold for (S x A + 1) / (S - 1) to be an integer K, which represents the number of coconuts that the last sailor stole from a pile of S x K + 1 coconuts. But this pile is the result of pushing together S - 1 equal piles by the previous thief, so again (S x K + 1) / (S - 1) is an integer and so on back through all S raids on the pile.

Note in the flowchart (and program) that the first trial value for A is 1 (Statement 70). In Statement 110 this value is increased by 1 until the value of (S x K + 1)/ (S - 1) is an integer as tested for in Statement 100. This process is then continued until the counter for the second loop, N (nighttime pile divisions) equals the number of sailors.

Although the program will work for any number of sailors, it takes a fairly long time to run for more than five. Remembering what you have learned earlier in this chapter, can you devise a way to make the program more efficient?


INPUT
S
A=1
K=A
N=O
LOOP
1
S*K+1
S-1
AN
INTEGER?
No
A=A+1
Yes
LOOP
2
N=N+1
K= S*K+1
S-1
No
N=S?
Yes
PRINT
S, S*K+1, A

```
10 PRINT "PROGRAM SOLVES SAI
LORS AND"
20 PRINT "MONKEY PROBLEM BY
WORKING"
30 PRINT "BACKWARDS."
40 PRINT
50 INPUT "NUMBER OF SAILORS=
":S
60 PRINT
70 A=1
80 K=A
90 N=0
100 IF (S*K+1)/(S-1)=INT((S*
K+1)/(S-1))THEN 130
110 A=A+1
120 GOTO 80
130 N=N+1
140 K=(S*K+1)/(S-1)
150 IF N=S THEN 170
160 GOTO 100
170 PRINT "THE FEWEST COCONU
TS THAT";S
180 PRINT "SAILORS CAN HAVE
TO BEGIN"
190 PRINT "IS";S*K+1
200 PRINT
210 PRINT "IN THE MORNING, E
ACH SAILOR"
220 PRINT "GETS";A
230 STOP
>
>
>RUN
PROGRAM SOLVES SAILORS AND
MONKEY PROBLEM BY WORKING
BACKWARDS.

NUMBER OF SAILORS=3

THE FEWEST COCONUTS THAT 3
SAILORS CAN HAVE TO BEGIN
IS 79

IN THE MORNING, EACH SAILOR
GETS 7

** DONE **

>RUN
PROGRAM SOLVES SAILORS AND
MONKEY PROBLEM BY WORKING
BACKWARDS.

NUMBER OF SAILORS=5

THE FEWEST COCONUTS THAT 5
SAILORS CAN HAVE TO BEGIN
IS 15621

IN THE MORNING, EACH SAILOR
GETS 1023

** DONE **
```

# Super Accuracy

Under normal circumstances, your computer performs computations to six or seven digits of accuracy. Double precision computations increase accuracy to 13 digits or so.

However, it is possible to do computations one digit at a time and assign each digit to an element in an array. This will achieve virtually any desired accuracy. Any, up to the maximum array size that is.

This example program performs the rather simple operation of successively doubling a number (which is the same as raising 2 to a power).

If the number to be represented is 8192, then:

| A(4) | A(3) | A(2) | A(1) |
|---|---|---|---|
| 8 | 1 | 9 | 2 |

To add this to itself, first the rightmost digits are added: A(1) + A(1). If there is a carry, the variable C is set equal to 1, otherwise C is 0. The total is put into B in Line 100.

If B is less than 10, there is no carry (C=0) and the new A(1) equals B. If B is greater than 10 there is a carry (C=1) and the new A(1) equals B-10.

This operation is continued for all the digits (D) of the number and, when it is finished, the new number is printed in Lines 220-240.

If A(N) is printed followed by a semicolon(;) for tight packing, the Basic print routine would leave a space in front of each digit (for the sign) and a space after each digit (for readability). In the program here, these spaces are not wanted, hence the print routine in Line 230 is used which prints the string value of the ASCII value of each digit (which is the same as the digit itself) but without the spaces.

This program, incidentally solves the challenge to calculate the number of moves in the Towers of Brahma problem (see "Change for Any Amount to $5.00). The approach is also used in the next section, "Palindromes."

```
10 PRINT "COMPUTES 2 TO NTH
TO ANY":"ACCURACY.":
20 DIM A(100)
30 M=0
40 C=0
50 D=0
60 A(1)=1
70 I=0
80 C=0
90 M=M+1
100 I=I+1
110 B=A(I)+A(I)+C
120 IF B>9 THEN 150
130 C=0
140 GOTO 170
150 B=B-10
```

```
160  C=1
170  A(I)=B
180  IF  I<D  THEN  100
190  IF  C=1  THEN  100
200  PRINT  M;
210  D=I
220  FOR  N=I  TO  1  STEP  -1
230  PRINT  CHR$(A(N)+48);
240  NEXT  N
250  PRINT
260  GOTO  70
```

```
>RUN
 COMPUTES 2 TO NTH TO ANY
 ACCURACY.

  1   2
  2   4
  3   8
  4   16
  5   32
  6   64
  7   128
  8   256
  9   512
  10   1024
  11   2048
  12   4096
  13   8192
  14   16384
  15   32768
  16   65536
  17   131072
  18   262144
  19   524288
  20   1048576
  21   2097152
  22   4194304
  23   8388608
  24   16777216
  25   33554432
  26   67108864
  27   134217728
  28   268435456
  29   536870912
  30   1073741824
  31   2147483648
  32   4294967296
  33   8589934592
  34   17179869184
  35   34359738368
  36   68719476736
  37   137438953472

  52   4503599627370496
  53   9007199254740992
  54   18014398509481984
  55   36028797018963968
  56   72057594037927936
  57   144115188075855872
  58   288230376151711744
  59   576460752303423488
  60   1152921504606846976
  61   2305843009213693952
  62   4611686018427387904
  63   9223372036854775808
  64   18446744073709551616
  65   36893488147419103232
  66   73786976294838206464
  67   147573952589676412928
  68   295147905179352825856
  69   590295810358705651712
  70   1180591620717411303424
  71   2361183241434822606848
  72   4722366482869645213696
```

# Palindromes

A palindrome is a word, verse, or number that reads the same backwards or forwards. For example, the words "mom" and "eye" are palindromes. So are each of the lines in this verse:

Egad, a base life defiles a bad age
Doom an evil deed, liven a mood
Harass sensuousness, Sarah
Golf; No, sir, prefer prison-flog
Ban campus motto, "Bottoms up, MacNab"

Numeric palindromes are those numbers which read the same backward as forward. The examination of these numbers is a field rich with possibilities for creative computing.

One conjecture concerning palindromes raises an interesting unanswered question. Begin with any positive integer. If it is not a palindrome, reverse its digits and add the two numbers. If the sum is not a palindrome, treat it as the original number and continue. The process stops when a palindrome is obtained. For example, beginning with 78:

$$\begin{array}{r} 78 \\ +\ 87 \\ \hline 165 \\ +561 \\ \hline 726 \\ +\ 627 \\ \hline 1353 \\ +3531 \\ \hline 4884 \end{array}$$

The conjecture, often assumed true, is that this process will always lead to a palindrome. And indeed that is just what usually happens. Most numbers less than 10,000 will produce a palindrome in less than 24 additions. But there is a real thorn in the side of this conjecture, the number 196. Can you determine if a palindrome will ever be produced with a starting number of 196?

The number 196 will produce 1675 after two reversals, but after 100 reversals the resultant sum has 47 digits and is still not palindromic. Why mention 1675? Because ten other numbers under 1000 will also lead to the sum of 1675 and thus may not become palindromic. The first five of these numbers are 196, 295, 394, 493, and 592. What are the other five?

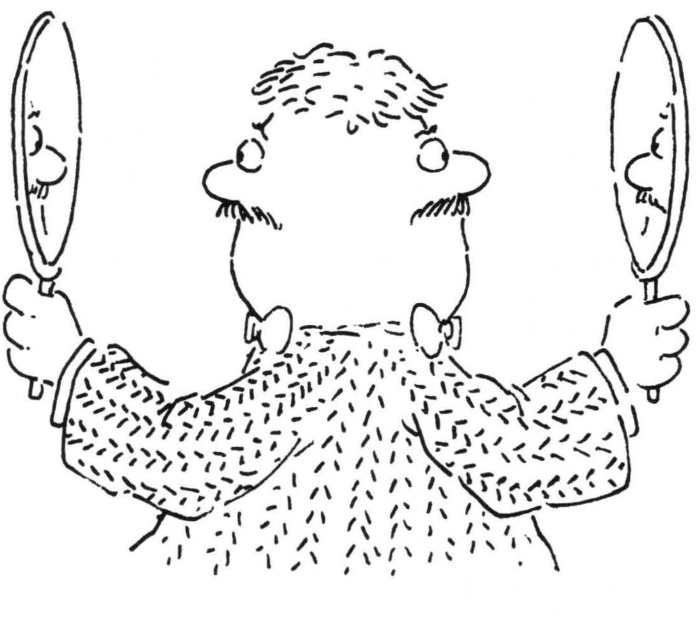
The program here will accept any number as a starting value and complete the process of adding the successive reversals and testing if the sum is a palindrome. Try it with some numbers and see if you can identify any patterns.

```
10 PRINT "PROGRAM TESTS CONJ
ECTURE"
20 PRINT "THAT REVERSING DIG
ITS AND"
30 PRINT "ADDING WILL ALWAYS
 PRODUCE"
40 PRINT "A PALINDROME."
45 PRINT "ENTER 0 (ZERO) TO
QUIT"
50 DIM B(50)
60 PRINT
70 INPUT "START NUMBER =":A
75 IF A=0 THEN 440
77 PRINT
80 E=0
90 E=E+1
100 A=A/10
110 IF INT(A)>0 THEN 90
120 FOR C=E TO 1 STEP -1
130 A=A*10
140 B(C)=INT(A-10*INT(A/10))
150 NEXT C
160 D=0
170 FOR C=1 TO INT(E/2)
180 IF B(C)=B(E+1-C)THEN 200
190 D=1
200 NEXT C
210 FOR C=E TO 1 STEP -1
220 PRINT CHR$(B(C)+48);
230 NEXT C
240 IF D=1 THEN 270
250 PRINT " IS A PALINDROME"
260 GOTO 60
270 PRINT " IS NOT A PALINDR
OME"
280 IF E/2<=INT(E/2)THEN 300
290 B(INT(E/2)+1)=2*B(INT(E/
2)+1)
300 FOR C=1 TO INT(E/2)
310 B(C)=B(C)+B(E+1-C)
320 NEXT C
330 FOR C=1 TO INT(E/2)
340 B(E+1-C)=B(C)
350 NEXT C
360 B(E+1)=0
370 FOR C=1 TO E
380 B(C+1)=B(C+1)+INT(B(C)/1
0)
390 B(C)=B(C)-10*INT(B(C)/10
)
400 NEXT C
410 IF B(E+1)<=0 THEN 160
420 E=E+1
430 GOTO 160
440 END
```

```
>RUN
PROGRAM TESTS CONJECTURE
THAT REVERSING DIGITS AND
ADDING WILL ALWAYS PRODUCE
A PALINDROME.
ENTER 0 (ZERO) TO QUIT

START NUMBER =19

19 IS NOT A PALINDROME
110 IS NOT A PALINDROME
121 IS A PALINDROME

START NUMBER =96

96 IS NOT A PALINDROME
165 IS NOT A PALINDROME
726 IS NOT A PALINDROME
1353 IS NOT A PALINDROME
4884 IS A PALINDROME

START NUMBER =0
```

Using this method, write a program that examines all the integers between 1 and 10,000 excluding those that sum to 1675 at any point. What does this show? By the way, you will have to devise a way to deal with 14-digit integers which are larger than your computer can normally handle.

Huh? Is this program "too hot to hoot?"



# 4

# Convergence and Recursion

The computer is especially suitable for doing repetitive and tedious calculations. Two mathematical approaches for solving problems that involve repetitive calculations are convergence and recursion.

Some problems can be reasonably easily stated in words or described with a few simple equations but there are many possible solutions. For example, how many ways can you make change for a dime? It is simply stated and the number of ways can be enumerated fairly easily: two nickels, one nickel and five pennies, or ten pennies, three ways in all. But if you want to solve for all the ways of making change for a dollar or five dollars, it would be nice to have some help.

Help on this kind of problem comes from a class of computer program that simply breaks the problem into smaller ones and counts up all the alternative solutions according to a set of rules. But an even more powerful technique is known as recursion. Using this technique, a simple solving algorithm or routine is set up to solve the smallest subset of the problem. The unique power in a recursive routine comes from the ability of the routine actuallly being able to call itself. This is discussed further in the second program in this section.

Another approach for solving problems that do not have an exact answer is that of successive approximations. For example, the exact value of pi, *e* or the length of an irregular curve cannot be precisely determined. But by means of increasingly accurate approximations, it is possible to approach the desired value from above or below or to converge on it from two directions. The last four programs in this chapter illustrate successive approximations and convergence.

# Change For a Dollar

Even though there is very little you can buy for a penny these days, the coin will probably be around for some time to come since it is needed to make change for odd amounts of sales tax and to fill up penny collections.

Today U.S. coinage consists of five coins: penny, nickel, dime, quarter, and half dollar. How many ways can coins of these denominations be used to make change for one dollar? For example, one way is two half dollars, another is one half dollar and two quarters, and so on. Make a best guess now and write it down before you read further.

There are several different ways to approach a problem of this kind. One is to break it down into smaller, more easily solved problems. In other words, how many ways can you make change for a quarter? For a dime? You would solve these subproblems and combine the answers to give the overall solution.

If you were more mathematically inclined, you could write a series of equations relating each piece of change to every other one and to the dollar and solve them.

A third approach is to do the problem by writing down combinations until all the different possibilities are exhausted (or until you are exhausted) and then count them all up. This might be called solving the problem by exhaustion and is a method quite suitable for putting on the computer.

Write a program that uses this approach to solve the problem. If you use loops and count by one, it could take a long time for the computer to run through all the possible combinations, possibly many hours.

Also, if you want to print out all the possible combinations, be warned that the printing could also take quite some time and a fair amount of paper. There are more combinations than you might think!

In fact, most people will not be able to guess the answer to this problem, or even come close. Ask several of your friends how many ways they think a dollar can be changed. Record all the responses and then tabulate them on your computer. What is the mean (average) of all the guesses? The extremes?

The program included here uses the first method discussed to solve the problem, in particular, breaking down the problem into subproblems and then combining the solutions into one final answer.

First, the main problem is broken into the next smaller one of making change for half dollars. There are three such problems: no half dollars (H = 0), one half dollar (H = 1), and two half dollars (H = 2). The last problem is trivial since there is only one way, but the other two need to be broken down further.

This is done by dividing the remaining money into quarters and considering the subproblems on down to the lower denominations. As the number of subproblems is expanded, each one becomes easier to solve. In fact,

subgoals, which can be solved in only one way, are finally reached. For example, if H = 1, Q = 1, D = 2, and N = 0, then the pennies (P) must equal five in order that the total equal 100.

Notice that at the quarter, dime and nickel stages, adjustments are made in the limits of the loops depending upon how much money there is left to change. For example, if H = 1, the only possible subgoals for quarters are 0, 1, and 2, but not 3 or 4. Also notice that there is no need to test combinations of coins to see if they add up to 100, nor is it necessary to include the penny as a variable. Simply counting the number of subgoals is sufficient since each one can be solved in only one way.

```
10 C=0
20 FOR H=0 TO 2
30 FOR Q=0 TO 4-2*H
40 FOR D=0 TO 10-5*H-.5*Q
50 FOR N=0 TO 20-10*H-5*Q-2*
D
60 C=C+1
70 NEXT N
80 NEXT D
90 NEXT Q
100 NEXT H
110 PRINT "THERE ARE";C;"DIF
FERENT WAYS";"TO CHANGE A DO
LLAR BILL."
```

```
>RUN
THERE ARE 292 DIFFERENT WAYS
TO CHANGE A DOLLAR BILL.

** DONE **
```

Try to make some changes in this program or write a new one to solve the following problems. Say you want two quarters in your change to play some video games. In how many ways can a dollar be changed to provide at least two quarters?

Visiting a small town, you find the parking meters still take pennies. In how many ways might you get change so that you had at least three pennies? Is this any different than the number of ways that would give you five pennies? Say you want to make a phone call also; in how many ways can you change a dollar to produce at least four pennies and one dime?

## Change For Any Amount to $5.00

Another way to attack the change problem in the previous section is by means of the programming technique called recursion. Get familiar with this one—it is very powerful! Donald Piele and Larry Wood described this method in an issue of *Creative Computing.*

First, define the variables which represent the number of ways to make change for n cents using the coins specified:

A Only pennies
B Nickels and pennies
C Dimes, nickels, and pennies
D Quarters, dimes, nickels, and pennies
E Halves, quarters, dimes, nickels, and pennies.

Initially, there are two subproblems in making change for n cents. In the first, no half dollars are used, and D is the number of ways to change n cents. Second, when one or more halves are used, after one is paid, there remain n–50 cents to pay which can be done in $E_{n-50}$ ways.

Since these two cases are mutually exclusive, it can be inferred that $E_n = D_n + E_{n-50}$. Similarly,

$D_n = C_n + D_{n-25}$
$C_n = B_n + C_{n-10}$
$B_n = A_n + B_{n-5}$

Now, begin with the simplest case and build up to $E_{100}$. First of all, it is easy to understand why $E_0 = 1$. From above, when $n = 50$, $E_{50} = D_{50} + E_0$, and it is possible to make change for 50 cents only one more way if half dollars are allowed. Therefore $E_0 = 1$. Likewise, $D_0 = C_0 = B_0 = A_0 = 1$. It is also true that $A_n = 1$ for all values of n since there is only one way to make change using only pennies. Now the recursive relationships can be used to solve the original problem.

This is the strategy used in the program. It also has the added advantage that it can count the number of ways of making change (with coins) for any specified amount.

Now it is your turn. Can you modify the program here to include dollar bills so it could count the number of ways to make change for any amount up to $10.00?

Using any method of change making you prefer, write a program to make change for one ruble. Russian coins come in denominations of 1, 2, 3, 5, 10, 15, 20, 50, and 100 kopecks. There are 100 kopecks in one ruble.

```
10 PRINT "PROGRAM COMPUTES W
AYS TO":"MAKE CHANGE FOR ANY
 AMOUNT":"UP TO $5.00"
20 PRINT
30 DIM A(101)
40 DIM B(101)
50 DIM C(101)
60 DIM D(101)
70 DIM E(101)
80 INPUT "AMOUNT TO CHANGE =
":X
90 M=INT(20*X)+1
100 A(1)=1
110 B(1)=1
120 C(1)=1
130 D(1)=1
140 E(1)=1
150 FOR J=2 TO M
160 A(J)=1
170 B(J)=A(J)+B(J-1)
180 C(J)=B(J)
190 IF J<=2 THEN 210
200 C(J)=B(J)+C(J-2)
210 D(J)=C(J)
220 IF J<=5 THEN 240
230 D(J)=C(J)+D(J-5)
240 E(J)=D(J)
250 IF J<=10 THEN 270
260 E(J)=D(J)+E(J-10)
270 NEXT J
280 PRINT
290 PRINT "YOU CAN MAKE CHAN
GE FOR":"$";X;"IN";E(M);"DIF
FERENT WAYS"
300 STOP
```

```
>RUN
PROGRAM COMPUTES WAYS TO
MAKE CHANGE FOR ANY AMOUNT
UP TO $5.00

AMOUNT TO CHANGE =4.99

YOU CAN MAKE CHANGE FOR
$ 4.99 IN 57420
DIFFERENT WAYS

** DONE **

>RUN
PROGRAM COMPUTES WAYS TO
MAKE CHANGE FOR ANY AMOUNT
UP TO $5.00

AMOUNT TO CHANGE =1.13

YOU CAN MAKE CHANGE FOR
$ 1.13 IN 384 DIFFERENT WAYS

** DONE **
```

Perhaps the most famous problem used to demonstrate the principles of recursion is the Towers of Brahma. It is sometimes called the Towers of Hanoi or Pharoah's Needles. Here is the problem in as close to original form as possible. You should be able to solve it with a relatively short program using recursion.

In the great temple at Benares beneath the dome which marks the center of the world rests a brass plate in which are fixed three diamond needles, each a cubit high and as thick as the body of a bee. On one of these needles, at the Creation, God placed 64 discs of pure gold, the largest disc resting on the brass plate and the others getting smaller and smaller up to the top one. This is the Tower of Brahma.

Day and night unceasingly, the priests transfer the discs from one needle to another, according to the fixed and immutable laws of Brahma. These laws require that the priest on duty must not move more than one disc at a time and that he must place this disc on a needle so there is no smaller disc below it. When the 64 discs shall have been thus transferred from the needle which, at the Creation, God placed them, to one of the other needles; tower, temple, and Brahmans alike will crumble into dust, and with a thunderclap, the world will vanish.

If the priests were to effect one transfer every second, and work 24 hours per day for each day of the year, it would take them 58,454,204,609 decades plus slightly more than six years to perform the feat, assuming they never made a mistake—for one small slip would undo all the work.

How many transfers are required to fulfill the prophecy? Try out your program with fewer discs than 64 to make sure you are on the right track. Here is a table of the first few transfers:

| *Discs* | *Moves* | *Discs* | *Moves* |
|---|---|---|---|
| 1 | 1 | 6 | 63 |
| 2 | 3 | 7 | 127 |
| 3 | 7 | 8 | 255 |
| 4 | 15 | 9 | 511 |
| 5 | 31 | 10 | 1023 |

# Converge on e and Pi

An incredibly important mathematical constant is designated by the small letter *e*. This constant is both irrational and transcendental. Look up those terms in a dictionary or math book if you wish, or just plunge on to the next paragraph.

The constant *e* was first derived by John Napier, also the inventor of logarithms, to whom we owe an eternal debt of gratitude. Why? If *e* had never been discovered, advances in mathematics, physics, and astronomy would have lagged a century or more, because *e* is the base of all natural logarithms and these logarithms are the basis for many branches of science and mathematics.

How is *e* calculated? The constant *e* is the limiting value of this expression as n approaches infinity:

$$e = (1 + {}^1\!/\!_n)^n$$

Its exact value can never be found, but to 15 places *e* equals 2.718281828459045 . . . How can *e* be calculated? First take 1-½ and square it; that equals 2-¼. Then cube 1-⅓ and you get 2.3686. Raising 1-¼ to the fourth power gives 2.414, and so on. Write a program for this method and have it print out the initial value of *e* and each value after each additional fraction is added on.

Another approach is to expand the expression above using the binominal theorem and, again, letting n approach infinity. The expression for the expansion is:

$$e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} \cdots \frac{1}{n!}$$

Now here is where the computer can again be of some assistance. Since 3! is 3*2! and 4! is 4*3!, all the calculations need not be done for each additional fraction. Look at the program and particularly note the calculations in Statements 70 and 90.

```
10 PRINT "CONVERGE ON E":"HI
T 'FCTN/4' TO HALT"
20 E=1
30 I=0
40 I=I+1
50 D=1
60 FOR J=1 TO I
70 D=D*J
80 NEXT J
90 E=E+1/D
100 PRINT E
110 GOTO 40
```

```
2
2.5
2.666666667
2.708333333
2.716666667
2.718055556
2.718253968
2.71827877
2.718281526
2.718281801
2.718281826
2.718281828
2.718281828
```

Pi is another important mathematical constant that is irrational (meaning its exact value can never be determined) and transcendental (meaning it is not the solution to any algebraic equation). Interestingly, pi was known and used by the ancients. Archimedes, who lived in the second century B.C., by using a regular polygon of 96 sides (nearly a circle), proved that the value of pi was less than 22/7 and greater than $3\text{-}^{10}/_{71}$, a remarkable achievement for the mathematics of his day.

Ptolemy in 150 A.D. used the value of 3.1416 for pi and in the middle of the sixteenth century the amazing fraction 355/113 was discovered, giving the value of pi accurately to six decimal places.

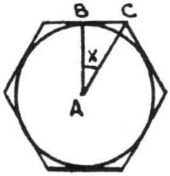
Incidentally, in 1897, the General Assembly of Indiana passed a bill ruling that the value of pi was four.

Several infinite series can be used to grind out increasingly accurate values for pi. One such series is $(1\text{-}^1/_3 + {}^1/_5 - {}^1/_7 + {}^1/_9 \ldots)$. This series is called an arithmetic series and converges very slowly. The program here displays only every 500th value of the series. Don't be alarmed if the program does not seem to be running very fast; a fair amount of calculating is going on between each value that is printed.

```
10 PRINT "CONVERGE ON PI WIT
H AN":"INFINITE SERIES"
20 PRINT "BE PATIENT; THIS I
S S-L-O-W"
30 PRINT "HIT 'FCTN/4' TO HA
LT":" "
40 S=1
50 I=1
60 Q=0
70 P=0
80 Q=Q+1
90 P=P+S/
100 I=I+2
110 S=-S
120 IF Q<499 THEN 80
130 Q=0
140 PRINT P*4
150 GOTO 80
```

```
CONVERGE ON PI WITH AN
INFINITE SERIES
BE PATIENT; THIS IS S-L-O-W
HIT 'FCTN/4' TO HALT

3.14359666
3.14059065
3.142260656
3.141091652
3.141993455
3.141258652
3.14187894
3.141342153
3.141815321
3.141392253
3.141774836
3.141425653
3.141746808
3.14144951
3.141726254
```

Actually, the approach used by Archimedes converges much more quickly and, with the aid of a computer, it is possible to go far beyond the 96-sided polygon used by Archimedes.




His approach was to construct inscribed and circumscribed polygons and measure the perimeters to approximate the circumference of a circle. Consider a polygon circumscribed around a circle of radius 1. The perimeter equals the length of one side times the number of sides. Since the tangent of x = AB/BC, but BC = 1, then tan (x) = AB and the length of a side = 2 tan (x). Since the circumference = 2 pi r and r = 1, then pi is the circumference (or perimeter of an n-sided polygon) divided by 2.

Similar trigonometry leads to the perimeter of an inscribed polygon being equal to the number of sides times sin (x) * cos (x).

The second program produces values for pi using inscribed and circumscribed polygons. Unfortunately, there is one large flaw in the program, because degrees must be converted into radians in Statement 50. This means, of course, that you must already know the value of pi, since the conversion factor is 360 degrees divided by 2 pi.

Setting this flaw aside, it is interesting to note how quickly this program converges on the value of pi compared to the preceding one. That is because this one converges geometrically rather than arithmetically.

Can you figure out a geometric convergence to the value of pi that does not require that you know it (or a conversion factor) before you start?

```
10 PRINT "CONVERGE ON PI":"H
IT 'FCTN/4' TO HALT":"
20 PRINT "INSCRIBED    CIRCU
MSCRIBED"
30 N=6
40 N=2*N
50 X=360/(N*57.29578)
60 PRINT N*SIN(X)*COS(X)/2,N
*TAN(X)/2
70 GOTO 40

CONVERGE ON PI
HIT 'FCTN/4' TO HALT

INSCRIBED         CIRCUMSCRIBED
 2.598076198       3.46410158
 2.999999977       3.215390281
 3.105828515       3.159659915
 3.132628587       3.146086188
 3.139350176       3.142714573
 3.141031924       3.141873023
 3.141452446       3.14166272
 3.141557581       3.14161015
 3.141583865       3.141597008
 3.141590437       3.141593722
```

# Convergence on Pi Revisited

In answer to the question posed in the last paragraph of the preceding section, here is a way to converge on pi without knowing its value beforehand.

As in the previous program, the basic approach is to add up the length of the sides on an inscribed polygon and divide by 2r to obtain a value for pi. The program starts with a square (four sides) and doubles the number of sides each time. If the old side length is S, then the length of S' of a side of a new polygon with twice as many sides is obtained by applying the Pythagorean theorem. In particular,

$$X^2 + (S/2)^2 = R^2$$

$$(R-X)^2 + (S/2)^2 = (S')^2$$

Thus,

$$S' = \sqrt{(R-\sqrt{R^2 - (S/2)^2})^2 + (S/2)^2}$$

It is easy to reduce this formula algebraically, but accuracy suffers if this is done. Also, you will find that S*S is slightly more accurate than S↑2.

Unfortunately, striving for maximum accuracy is somewhat moot on a computer that does not have double precision arithmetic. Notice that accuracy does not improve with more than 8192 sides and, indeed, as numbers in the calculations start to exceed the capacity of the computer, the accuracy starts to deteriorate badly.

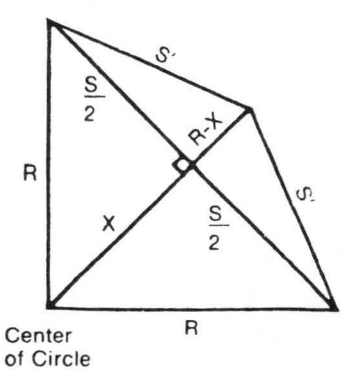
There is yet another method to compute pi by convergence. It uses discoveries of Gregory and Euler. Gregory discovered the formula for arctangent:

$$\text{Arctan } x = x - x^3/3 + x^5/5 - x^7/7 + \ldots$$

Euler came up with a rather interesting formula for pi:

$$\text{pi} = 4\,(\arctan\,(1/2) + \arctan\,(1/3))$$

See if you can combine these two formulas to calculate pi. If you are very clever, you can do your calculation to yield far more than the seven decimal place accuracy obtained by the programs presented here so far.



```
10 PRINT "CONVERGE ON PI WIT
H":"INSCRIBED POLYGONS":" "
20 PRINT "WHEN PROGRAM BREAK
S, TYPE":"'CON' TO CONTINUE"
:" "
30 PRINT "SIDES","PERIMETER"
40 R=10
50 Z=2
60 N=Z+Z
70 S=R*SQR(Z)
80 FOR K=1 TO 28
90 IF K<>14 THEN 110
100 BREAK 110
110 PRINT N,(S*N)/(Z*R)
120 Y=S*S/(Z+Z)
130 X=R-SQR(R*R-Y)
140 S=SQR(X*X+Y)
150 N=N*Z
160 NEXT K
```

```
>RUN
CONVERGE ON PI WITH
INSCRIBED POLYGONS

WHEN PROGRAM BREAKS, TYPE
'CON' TO CONTINUE

SIDES               PERIMETER
 4                  2.828427125
 8                  3.061467459
 16                 3.121445152
 32                 3.136548491
 64                 3.140331157
 128                3.141277251
 256                3.141513801
 512                3.14157294
 1024               3.141587725
 2048               3.141591422
 4096               3.141592346
 8192               3.141592577
 16384              3.141592634

* BREAKPOINT AT 100
>
>CON
 32768              3.141592649
 65536              3.141592652
 131072             3.141592653
 262144             3.141592654
 524288             3.141592654
 1048576            3.141592654
 2097152            3.141592654
 4194304            3.141592654
 8388608            3.141592654
 16777216           3.141592654
 33554432           3.141592654
 67108864           3.141592654
 134217728          3.141592654
 268435456          3.141592654
 536870912          3.141592654

** DONE **
```

# Length of Any Curve

The previous programs have demonstrated how it is possible to compute a very accurate value of pi by adding together the length of the sides of a polygon as it approaches a circle and dividing by 2r.

Using a similar approach, it should be possible to inscribe a polygon, or portions of a polygon, inside any regular curve and thus determine the length of the curve. This program approximates the length of any curve as defined in Statement 110 by dividing it into an increasing number of subintervals and computing the sum of the secants (a straight line that cuts a curve at two or more points).

To run the program, you must enter the formula or equation describing your curve in Statement 110 in the form:

DEF FNA(X) = Any function of X

You then type RUN and enter the end points of the curve you want to use in your calculation. These points are entered in the form of the abscissa, which means the horizontal (or y) coordinate of the point.

The program is written to sum the successive secant lengths and to calculate the percent of change in each summation compared to the preceding one. The sample run uses the function $2x^3 + 3x^2 - 2x + 3$. Note the substantial improvements in the length calculation as the number of intervals increases from 2 to 16, but the rather slight improvements beyond 16.

Try this program with different curves and functions. It might help to plot the function first (remember the program to do that?) and then compute its length. Is this method of length calculation more accurate for a function with no changes in direction of the curve within the interval selected or for a function with one or more changes? Why?

```
10 PRINT "CURVE LENGTHS":" "
20 PRINT "DEFINE YOUR FUNCTI
ON IN":"LINE 110"::
30 PRINT "ENTER ABSCISSAS OF
 END PTS":"(SMALLER ONE FIRS
T)";
40 INPUT P,Q
50 PRINT " ":"NO. OF   SUM OF
      PERCENT"
60 PRINT "SUBIN-   SECANT
CHANGE IN"
70 PRINT "TERVALS LENGTHS
LENGTH"
80 PRINT
90 S1=0
100 FOR N=1 TO 9
110 DEF FNA(X)=2*X^3+3*X^2-2
*X+3
120 E=2^(N-1)
130 H=(Q-P)/E
140 S=0
150 FOR I=0 TO E-1
160 L=SQR((FNA(P+I*H+H)-FNA(
P+I*H))^2+H*H)
170 S=S+L
180 NEXT I
190 IF S1>0 THEN 220
200 PRINT E;TAB(5);S;TAB(18)
;"NO PREV VAL"
210 GOTO 240
220 P5=((ABS(S1-S))/S1)*100
230 PRINT E;TAB(5);S;TAB(17)
;P5
240 S1=S
250 NEXT N
```

```
CURVE LENGTHS

DEFINE YOUR FUNCTION IN
LINE 110

ENTER ABSCISSAS OF END PTS
(SMALLER ONE FIRST)? -1,6

NO. OF   SUM OF      PERCENT
SUBIN-   SECANT      CHANGE IN
TERVALS  LENGTHS     LENGTH

 1      525.0466646 NO PREV VAL
 2      525.158262  .0212547633
 4      529.652247  .8557391718
 8      531.0171331
                    .2576947669
 16     531.9642557
                    .1783600797
 32     532.0165657
                    .0098333641
 64     532.0416782
                    .0047202447
```

# Converge on a Square Root

Since most square roots are irrational, methods used to calculate them usually involve successive approximations. Although you can simply call up the square root function in Basic or on many pocket calculators, it is interesting to explore various methods of calculating square roots without these built-in functions. After all, these built-in functions are nothing more than successive approximation routines already installed in the machine.

Obviously, a square root is the inverse of the operation of squaring a number. All of the methods of calculating square roots use this fact, but the way in which it is employed is quite different in various calculators and computers.

The program here calculates an upper and lower limit for the square root of a number and, by successive approximations, pinches the root to within a smaller and smaller interval until it reaches the desired level of accuracy.

The starting value for the lower limit is 0 and for the upper limit the number, Z, whose square root is sought. The program then divides this interval into ten steps by simply dividing the difference between the numbers by 10. The variable I is increased from the lower limit to the upper one by the value of the step, S. At any point, if I squared becomes greater than Z, a new upper limit is set to I and a new lower limit is set to I - S.

This method converges very quickly and adds approximately one decimal place of accuracy with each pass beyond the third. What happens when you enter into the program a number that has an exact square root such as 25 or 49? Why?

Another approach to calculating square roots by successive approximations is to start with a trial root, X. If X * X is less than the original number N, then increase the trial value by a 0.1. If X * X is greater than N, return to the previous value. This is the first digit of the root. Now, start advancing by 0.01. Continuing in this way, one digit is developed at a time until the desired precision is reached.

This method is quite suitable and fast for square roots of numbers less than 1. A good first trial root value is 0.1. But is it suitable for larger numbers? How should a starting trial value be determined? In this method, especially for numbers greater than 10, the initial trial value for the root matters a great deal in determining the length of time it will take for the calculation to converge. Write a program using this method and compare the speed and accuracy with the program in the book.

```
10 PRINT "CALCULATE SQUARE R
OOTS":" "
20 PRINT "SET ACCURACY IN LI
NE 40"
30 INPUT "YOUR NUMBER =":Z
40 E=.00001
50 PRINT " ":"LOWER LIMIT","
UPPER LIMIT"
60 A=0
70 B=Z
80 S=(B-A)/10
90 PRINT A,B
100 IF ABS(A*B-Z)<E THEN 190
110 FOR I=A TO B STEP S
120 IF Z<I*I THEN 160
130 NEXT I
140 B=B*10
150 GOTO 80
160 B=I
170 A=I-S
180 GOTO 80
190 PRINT " ":"ACCURACY LIMI
T REACHED"
200 PRINT "AVERAGE VALUE =";
(A+B)/2

>RUN
CALCULATE SQUARE ROOTS

SET ACCURACY IN LINE 40
YOUR NUMBER =3

LOWER LIMIT    UPPER LIMIT
 0              3
 1.5            1.8
 1.71           1.74
 1.731          1.734
 1.7319         1.7322

ACCURACY LIMIT REACHED
AVERAGE VALUE = 1.73205

** DONE **

>RUN
CALCULATE SQUARE ROOTS

SET ACCURACY IN LINE 40
YOUR NUMBER =54

LOWER LIMIT    UPPER LIMIT
 0              54
 5.4            10.8
 7.02           7.56
 7.344          7.398
 7.344          7.3494
 7.34832        7.34886
 7.348428       7.348482
 7.3484658      7.3484712
 7.34846904     7.34846958

ACCURACY LIMIT REACHED
AVERAGE VALUE = 7.3484693I

** DONE **
```

# 5

# Compounding

As with successive approximations and recursion, compounding requires many repetitive calculations. Some compound interest and growth situations can be represented by a formula, but for many problems, solving by repetitive calculations is an excellent approach.

There is nothing magical about compounding; you generally start with an initial amount of money, number of animals, etc. This quantity then grows or diminishes by a certain percentage at set intervals. The new amount is the old amount increased or decreased by this percentage. Repeat this calculation over and over again, and you have solved the problem.

Five different types of problems which involve compounding of some sort are described in this chapter. Try the extra problems that follow some of the programs; you may be surprised at the results.

# Indians and Interest

Here is a simple compound interest problem that produces an astonishing answer. The problem has to do with the sale of Manhattan Island to the Dutch for approximately $24 worth of trinkets and beads.

If the $24 that the Indians received in 1626 had been deposited in a bank paying 5-$^3/_4$% interest compounded annually, how much would it amount to in 1983?

The program here solves this little problem by making use of the formula to calculate compound interest. In particular, if P dollars are invested at an interest rate of R (expressed as a decimal) and compounded N times, then the total amount A is given by the formula:

$$A = P(1 + R)^N$$

How much was gained in 1983 alone? How much was gained in the decade from 1974 to 1983? You can change the value of N in Statement 30 to get the answers to these questions. But a better way might be to enter the ending year with an INPUT statement.

Can you read a number expressed in the E (exponential format)? In a more conventional format, the number is $11,176,500,000 or $11.2 billion.

There are many other ways you can improve the program and make it more suitable for general purpose compound interest calculations. Modify it to accept any starting and ending year, any rate of interest, and any starting principal amount.

The problem as stated is somewhat unrealistic since banks were not paying interest rates of 5-$^3/_4$% from 1626 to 1983. Change the program to calculate the total amount based on the following interest rates:

| *Years* | *Interest Rate* |
|---|---|
| 1626-1830 | 1.5% |
| 1831-1870 | 2.0% |
| 1871-1910 | 3.0% |
| 1911-1921 | 3.5% |
| 1922-1929 | 6.5% |
| 1930-1940 | 2.3% |
| 1941-1945 | 3.5% |
| 1946-1960 | 5.3% |
| 1961-1980 | 6.5% |
| 1981-1983 | 9.5% |

```
10 P=24
20 R=.0575
30 N=1983-1626
40 A=P*((1+R)^N)
50 PRINT "IF THE $24 WHICH T
HE INDIANS":"RECEIVED IN 162
6 FOR MAN-":"HATTAN ISLAND H
AD BEEN"
60 PRINT "INVESTED AT 5.75%,
 BY 1983,":"IT WOULD HAVE AM
OUNTED TO":"$";A
```

```
>RUN
IF THE $24 WHICH THE INDIANS
RECEIVED IN 1626 FOR MAN-
HATTAN ISLAND HAD BEEN
INVESTED AT 5.75%, BY 1983,
IT WOULD HAVE AMOUNTED TO
$ 1.11765E+10

** DONE **
```

# Systematic Savings

In the previous program, compound interest was calculated by means of a formula well known to bankers, money lenders and real estate agents. However, if you did not know the formula, how would you calculate interest on money in a savings account by hand or with a calculator?

You would probably begin by multiplying the principal amount P by the interest rate R and adding that amount to the original principal at the start of the second year. Doing that for all the years the money is in the bank will yield a final amount. Why not write a program to perform the calculations in this manner rather than use a formula? Here is such a program.

It is set up for an initial principal amount of 100 (Statement 50), an interest rate of 10% (R = 0.1 in Statement 60) and ten years (N = 10 in Statement 70). Naturally these values could be read in using INPUT statements. The clever calculation in Statement 100 rounds off the amount to two decimal places (dollars and cents).

In contrast with the program in the previous section, this one does not use a compound interest formula, but simply adds the interest each year to the growing principal amount in Statement 90.

```
10 PRINT "CALCULATES INTERES
T ON $100":"INVESTED AT 10%
FOR 10 YRS":"
20 PRINT "AT END","CURRENT"
30 PRINT "OF YEAR","BALANCE"
40 PRINT
50 P=100
60 R=.1
70 N=10
80 FOR I=1 TO N
90 P=P+P*R
100 PRINT I,INT(P*100+.5)/10
0
110 NEXT I

>RUN
CALCULATES INTEREST ON $100
INVESTED AT 10% FOR 10 YRS

AT END          CURRENT
OF YEAR         BALANCE

 1              110
 2              121
 3              133.1
 4              146.41
 5              161.05
 6              177.16
 7              194.87
 8              214.36
 9              235.79
 10             259.37
```

Now, how could this program be modified to allow for a plan of systematic saving? In other words, instead of letting the $100 lie around all lonely while it is compounding, each year you add another $100 to it. With this program, making the modification for systematic savings is easy: Statement 105 is added to add the new deposit each year to the ever growing principal.

```
10 PRINT "CALCULATES INTERES
T ON $100":"INVESTED AT 10%
EVERY YEAR":"FOR 10 YEARS":"
"
20 PRINT "AT END            C
URRENT"
30 PRINT "OF YEAR INVESTED B
ALANCE"
40 PRINT
50 P=100
55 C=100
60 R=.1
70 N=10
80 FOR I=1 TO N
90 P=P+P*R
100 PRINT I;TAB(11);I*C;TAB(
20);INT(P*100+.5)/100
105 P=P+C
110 NEXT I
>
>
>RUN
CALCULATES INTEREST ON $100
INVESTED AT 10% EVERY YEAR
FOR 10 YEARS

AT END              CURRENT
OF YEAR  INVESTED   BALANCE

 1          100        110
 2          200        231
 3          300        364.1
 4          400        510.51
 5          500        671.56
 6          600        848.72
 7          700        1043.59
 8          800        1257.95
 9          900        1493.74
 10         1000       1753.12

** DONE **
```

In an effort to attract depositors, many banks over the last 20 years have started offering savings and investment accounts in which the deposits are compounded more often than annually. In the early 50's, many banks went to quarterly compounding; in the late 50's, to daily; and in the early 60's, some "competitive" S&L's went to continuous compounding.

How much does more frequent compounding really mean? Modify either of the two programs to compute the interest for more frequent compounding. What is the difference in interest for one year for annual, quarterly, monthly, daily, and continuous (every second) compounding on a principal amount of \$1000 invested at 8%? How about for ten years?

Incidentally, if you want to use the formula method in the previous section for this calculation, the formula for P principal invested at R rate compounded N times per year is:

$$A = P(1 + R/N)^N$$

Try this problem which uses the same principles of compounding. Consumer prices rose an average of 8.8% in 1980. While the government keeps trying to bring inflation under control, they don't seem to be meeting with much success. Assuming that prices continue to go up this much (8.8%) every year, how much will a \$6000 economy car (1980 dollars) cost in the year 2000? How much will it cost when you are 65 years old?

# Systematic Savings Revisited

Using the formula for compound interest and systematic savings, this program will calculate the amount accumulated after a given period of time.

When you run the program, it will ask how much you wish to save each month, the number of compounding periods in a year, the interest rate, and the length of time you wish to continue your systematic savings program. The program will then calculate the total amount at the end of that period.

From the preceding programs, you should be able to see that a systematic savings program is a very effective way to accumulate a nest egg for the future.

```
10 PRINT "CALCULATES INTERES
T AND":"BALANCE FOR A SYSTEM
ATIC":"SAVINGS PROGRAM":" "
20 INPUT "MONTHLY DEPOSIT ="
:A
30 INPUT "COMPOUNDING PERIOD
S/YR =":B
40 INPUT "INTEREST RATE (XX.
X %) =":R
50 R=R/100
60 INPUT "NUMBER OF YEARS ="
:N
70 PRINT
80 PRINT "AT END           C
URRENT"
90 PRINT "OF YEAR INVESTED B
ALANCE"
100 PRINT
110 C=12*A
120 P=12*A
130 FOR I=1 TO N
140 P=P*(1+R/B)^B
150 PRINT I;TAB(11);I*C;TAB(
20);INT(P*100+.5)/100
160 P=P+C
170 NEXT I
```

```
>RUN
 CALCULATES INTEREST AND
 BALANCE FOR A SYSTEMATIC
 SAVINGS PROGRAM

 MONTHLY DEPOSIT =10
 COMPOUNDING PERIODS/YR =12
 INTEREST RATE (XX.X %) =15
 NUMBER OF YEARS =2

 AT END           CURRENT
 OF YEAR INVESTED BALANCE

  1          120      139.29
  2          240      300.97

 ** DONE **
```

```
CALCULATES INTEREST AND
BALANCE FOR A SYSTEMATIC
SAVINGS PROGRAM

MONTHLY DEPOSIT =10
COMPOUNDING PERIODS/YR =1
INTEREST RATE (XX.X %) =8.5
NUMBER OF YEARS =8

AT END                    CURRENT
OF YEAR  INVESTED         BALANCE

 1            120          130.2
 2            240          271.47
 3            360          424.74
 4            480          591.04
 5            600          771.48
 6            720          967.26
 7            840          1179.68
 8            960          1410.15

** DONE **

CALCULATES INTEREST AND
BALANCE FOR A SYSTEMATIC
SAVINGS PROGRAM

MONTHLY DEPOSIT =10
COMPOUNDING PERIODS/YR =365
INTEREST RATE (XX.X %) =8.5
NUMBER OF YEARS =8

AT END                    CURRENT
OF YEAR  INVESTED         BALANCE

 1            120          130.64
 2            240          272.88
 3            360          427.73
 4            480          596.32
 5            600          779.66
 6            720          979.58
 7            840          1197.23
 8            960          1434.08

** DONE **
```

Try to combine the things that you have learned from the programs in these sections to write a completely generalized program for systematic savings. It should accept the following information as input:

- Initial deposit
- Frequency of periodic deposits
- Amount of each deposit
- Interest rate
- Interest compounding frequency
- Length of time of savings program

Your program should produce output in tabular form showing years (or other time periods), amount invested without interest, total amount with interest, and the interest alone.

# Loan Payments

Although saving money in a systematic way is a noble goal, many people frequently find themselves talking to a different bank officer, namely the one in charge of loans.

This program will calculate the payments for a loan for a period of one year or longer. The program asks you to enter the key facets of the loan in question: amount borrowed, annual rate of interest, interval between payments and term of the loan in years.

The sample run shows a relatively short-term loan (2 years) of $3000 at a bargain basement interest rate of 8%. The monthly payment of the loan is computed to be $135.68 and the total interest $256.34. Why is the total interest not $240 (8% of $3000)? Instead it is 8.54% Is this a mistake in the program?

Try the program for some longer term loans at real world interest rates. For example, run it for an automobile loan of $8000 at 12% over a 5-year period. Perhaps you can see from this that systematically saving your money and paying cash for an item makes more sense than time payments.

Run the program to calculate the mortgage payments on a $150,000 house with a $40,000 down payment. Try it with a 16% interest rate stretched over a 30-year period. Ouch! Look at all that interest!

At the heart of this program are Statements 150-180. Can you see what is happening in these calculations?

```
>RUN
PROGRAM CALCULATES PAYMENTS
ON A LONG-TERM LOAN

AMOUNT BORROWED =125000
INTEREST RATE (XX.X %) =17
PAYMENT INTERVAL (MOS.) =1
TERM OF LOAN (YEARS) =25
OUTPUT AS TABLE (ENTER 1)
OR TOTALS ONLY ENTER 2)? 2

TOTAL INTEREST = 414153.61
PRINCIPAL REPAID = 125000
MONTHLY PAYMENT =$ 1797.18
AND TOTALS $ 539153.61

** DONE **
```

```
10 PRINT "PROGRAM CALCULATES
 PAYMENTS":"ON A LONG-TERM L
OAN":" "
20 INPUT "AMOUNT BORROWED ="
:A
30 INPUT "INTEREST RATE (XX.
X %) =":I
40 INPUT "PAYMENT INTERVAL (
MOS.) =":P
50 INPUT "TERM OF LOAN (YEAR
S) =":Y
60 PRINT "OUTPUT AS TABLE (E
NTER 1)":"OR TOTALS ONLY ENT
ER 2)";
70 INPUT V
80 PRINT
90 IF V=2 THEN 150
100 PRINT " ":"TABLE IN";P;"
MONTH INTERVALS":" "
110 PRINT TAB(6);"PRNCPL";TA
B(22);"PRNCPL"
120 PRINT "PER- AT BEG  INT
DUE  REPAID"
130 PRINT "IOD  OF PER  END
PER  END PER"
140 PRINT
150 Z=(Y*12)/P
160 K=(I*(P/12))/100
170 E=A*K/(1-1/(1+K)^Z)
180 E=INT(E*100+.5)/100
190 C=A
200 F=0
210 D1=0
220 T1=0
230 T1=T1+1
240 IF T1>Z THEN 360
250 B=T1
260 C=C-F
270 D=C*K
280 F=E-D
290 C=INT(C*100+.5)/100
300 D=INT(D*100+.5)/100
310 F=INT(F*100+.5)/100
320 D1=D1+D
330 IF V=2 THEN 230
340 PRINT B;TAB(5);C;TAB(14)
;D;TAB(22);F
350 GOTO 230
360 IF V=1 THEN 410
370 D1=INT(D1*100+.5)/100
380 PRINT "TOTAL INTEREST ="
;D1
390 PRINT "PRINCIPAL REPAID
=";A
400 GOTO 430
410 PRINT
420 PRINT "TOTAL";TAB(14);D1
;TAB(22);A
430 E5=INT((D1+A)*100+.5)/10
0
440 E6=E5/((Y*12)/P)
450 E6=INT(100*E6+.5)/100
460 PRINT "MONTHLY PAYMENT =
$";E6
470 PRINT "AND TOTALS $";E5
```

```
>RUN
PROGRAM CALCULATES PAYMENTS
ON A LONG-TERM LOAN

AMOUNT BORROWED =3000
INTEREST RATE (XX.X %) =8
PAYMENT INTERVAL (MOS.) =1
TERM OF LOAN (YEARS) =2
OUTPUT AS TABLE (ENTER 1)
OR TOTALS ONLY ENTER 2)? 2

TOTAL INTEREREST = 256.34
PRINCIPAL REPAID = 3000
MONTHLY PAYMENT =$ 135.68
AND TOTALS $ 3256.34

** DONE **

>RUN
PROGRAM CALCULATES PAYMENTS
ON A LONG-TERM LOAN

AMOUNT BORROWED =8000
INTEREST RATE (XX.X %) =13
PAYMENT INTERVAL (MOS.) =3
TERM OF LOAN (YEARS) =5
OUTPUT AS TABLE (ENTER 1)
OR TOTALS ONLY ENTER 2)? 1


TABLE IN 3 MONTH INTERVALS

       PRNCPL                  PRNCPL
PER-   AT BEG     INT DUE      REPAID
IOD    OF PER     END PER      END PER

 1     8000       260          290.23
 2     7709.77    250.57       299.66

 6     6451.41    209.67       340.56
 7     6110.85    198.6        351.66
 8     5759.22    187.17       363.06
 9     5396.16    175.38       374.85
 10    5021.31    163.19       387.04
 11    4634.27    150.61       399.62
 12    4234.65    137.63       412.6
 13    3822.05    124.22       426.01
 14    3396.04    110.37       439.86
 15    2956.18    96.08        454.15
 16    2502.03    81.32        468.91
 17    2033.12    66.08        484.15
 18    1548.97    50.34        499.89
 19    1049.08    34.1         516.13
 20    532.95     17.32        532.91

TOTAL             3004.64
                              8000
MONTHLY PAYMENT =$ 550.23
AND TOTALS $ 11004.64

** DONE **
```

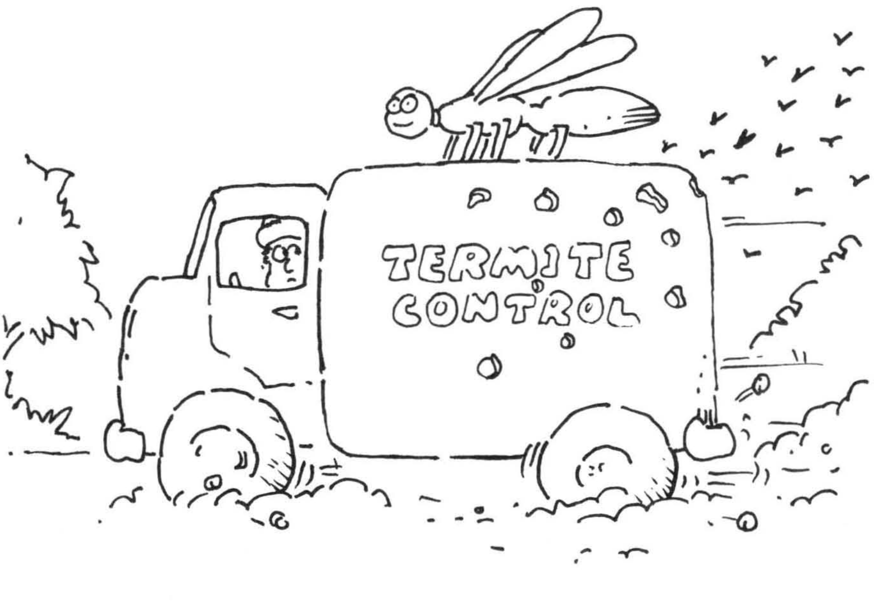
# Interest on Credit Purchases

All too frequently, the rate of interest to be charged on a loan or credit purchase is hidden in the small type. After all, the bank or car dealer or finance company wants to convince you that you can afford that new car or home improvement or whatever.

This program calculates the interest rate on a loan given the principal value of the loan, the number of payments, and the amount per payment. To make things even easier, the program will accept the cash purchase price of the article and the down payment and automatically compute the principal value of the loan.

The sample run shows a rather unrealistic loan for an item costing $88.99. The down payment was $10.00 and the loan is over a period of 18 months; each monthly payment is $4.85. The program run shows that the actual interest rate is a modest 5.69%.

Now try the program with a more realistic loan. A termite company in New Jersey advertises a total treatment for your house for only $200; just $29.95 down and 24 monthly payments of $11.95 per month. Is this the bargain that it seems? What is the annual rate of interest?



```
10 PRINT "PROGRAM CALCULATES
 INTEREST":"RATE ON CREDIT P
URCHASES":"
20 INPUT "PRICE OF ITEM =":C
30 INPUT "DOWN PAYMENT =":D
40 INPUT "NUMBER OF PAYMENTS
 =":N
50 INPUT "PAYMENTS PER MONTH
 =":S
60 INPUT "AMOUNT PER PAYMENT
 =":R
70 B=R*N+D
80 I=B-C
90 M=N/(S*12)
100 T=INT(100*(I*100/(B*M))+
.5)/100
110 PRINT " ":"INTEREST RATE
 IS";T;"%"


>RUN
 PROGRAM CALCULATES INTEREST
 RATE ON CREDIT PURCHASES

 PRICE OF ITEM =88.99
 DOWN PAYMENT =10
 NUMBER OF PAYMENTS =18
 PAYMENTS PER MONTH =1
 AMOUNT PER PAYMENT =4.85

 INTEREST RATE IS 5.69 %

 ** DONE **
```

# Population Growth

So far, all the compounding problems in this chapter have involved money—interest, savings, and loans. But many fascinating compounding problems do not involve money. Consider the following problem.

In 1960, the population figures for the United States and Mexico were 180 million and 85 million respectively. The annual population growth rate for the United States was 1.23% and for Mexico 2.23%. If these growth rates remain steady, in some distant year the population of Mexico will exceed that of the United States. In what year will this occur?

The program uses the method of accumulating the principal amount rather than a formula, and applies it to both populations. The sample run reveals that the population of Mexico will overtake that of the U.S. in the not too distant future.

```
10  U=180000000
20  M=85000000
30  R1=1.0123
40  R2=1.0223
50  Y=1960
60  U=U*R1
70  M=M*R2
80  Y=Y+1
90   IF  M<U  THEN 60
100  PRINT "YEAR =";Y;"     POP
ULATION":"
110  PRINT "U.S.A.",U
120  PRINT "MEXICO",M
>
>
>RUN
YEAR  =  2037          POPULATION

U.S.A.                 461406536.3
MEXICO                 464464693.9
```

But what if the Mexican people were diligently trying to bring their increasing population under control and their annual growth rate was increasing by only 0.001% per year, compared to the increase of the U.S. growth rate of 0.01% per year. If that were the case, would the population of Mexico ever exceed that of the U.S.?

Insert Statements 60, 70, and 85 into the first program, run it and find out the answer to the question.

```
>60  U=U*(1.01*R1)
>70  M=M*(1.001*R2)
>85  IF  Y>9999  THEN  100
```

Going back to the first program, run it with the population data from the U.S. (180 million) and California (15.7 million) from 1960. At that time, the annual growth rates were 1.23% and 3.7% respectively. Running the program will indicate the year that the population of California will exceed that of the United States. This, of course, is nonsense. Where is the discrepancy?

This program might make more sense if it were restated to ask in what year the population of California would exceed that of the remainder of the U.S., if ever?

Compounding can also be used to solve other kinds of population growth problems. Consider the bristleworm. The bristleworm can reproduce by splitting itself into 24 segments, each of which grows a new head and tail. What is the maximum number of bristleworms that could be obtained in this fashion, starting with only one worm, after ten splittings? Assuming a splitting occurs every 22 days, how many offspring will one worm produce in a year? When will the earth be overrun with bristleworms?

Here is another problem for which the principle of compounding is useful. It takes nature about 500 years to produce one inch of topsoil. Many years ago, the United States had an average depth of almost nine inches of this good dirt, but as of 1975, the country was down to about six inches. This type of soil is necessary, of course, for growing food.

Careless management of our soil causes about 1% per year to erode away; it is then lost forever. Once soil depth reaches three inches or less, it is impossible to grow crops on a large scale. Write a program to calculate the year in which the U.S. will have less the 3″ of topsoil, assuming that it continues to erode away at 1% per year.

Will you be alive then? Will your children be alive? Will anyone be alive?

# 6

# Probability

Statistics and probability are subjects that bring up all kinds of images. Some people who have not had a pleasant time in a required college statistics course keep as far away from the subject as possible. To other people, statistics is something with which devious manufacturers can mislead you about their products.

Not long ago, one foreign automobile manufacturer boasted that 90% of their cars sold in the United States in the last seven years were still on the road. This sounds like excellent reliability. But consider the fact that this manufacturer was rapidly expanding in the U.S. market and 65% of the cars they had sold in the U.S. had been sold in just the previous two years. You would expect that all of these would still be running.

If the manufacturer had sold 10% of their 7-year volume in Years 1 to 3, and most of these cars were not running, then the advertising claim loses much of its meaning.

When approached in a logical, step-by-step manner, statistics and probability are not difficult. In fact, they can be a great deal of fun.

Some of the programs use a formula while others simulate the event in which we are interested. The results are the same, but you may find that the simulations help you to understand exactly what is happening.

# Pascal's Triangle—Calculated

Pascal's Triangle is quite fascinating. As you can see from the portion of one reproduced below, each row is symmetrical. Each row also contains the coefficients for a binomial expansion. The sums across the ascending diagonals form the Fibonacci sequence. The sums across the rows are all powers of two. Each row corresponds to the digits of a power of 11. Every element is the sum of the two above it. And, in case you care, all the elements in it are identities in combinatorial theory.

The ways this marvelous triangle can be generated are as varied and interesting as its properties, though perhaps more difficult to figure out. Here is one way to use a computer to generate the triangle.

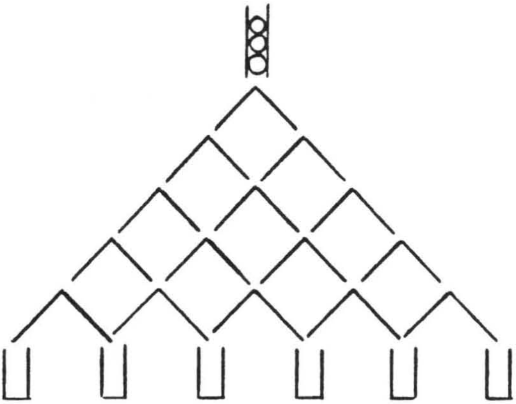
Any element can be found be adding together the two elements immediately above it. The program uses this principle to produce a triangle eight levels deep. Lines 10-30 set the rightmost diagonal to 1. Each element is stored in the two-dimensional variable P(R,C) with R denoting the row and C denoting the "column." The variable T (Line 50) simply leaves some blank spaces so the output resembles a triangle. The crux of the calculation is in Line 70 in which the value of each new element is calculated.

There is another interesting way to calculate Pascal's Triangle in even fewer statements than the program here. It generates the triangle one element at a time and does not use any arrays or two-dimensional variables. Can you figure out how to write such a program?

```
10 FOR C=1 TO 9
20 P(C,C)=1
30 NEXT C
40 FOR R=1 TO 9
50 T=17-R*2
60 FOR C=2 TO R+1
70 P(R+1,C)=P(R,C)+P(R,C-1)
80 IF P(R,C)=0 THEN 110
90 PRINT TAB(T);P(R,C);
100 T=T+4
110 NEXT C
120 PRINT
130 PRINT
140 NEXT R

>RUN

                  1

                1   1

              1   2   1

            1   3   3   1

          1   4   6   4   1

        1   5   10  10  5   1

      1   6   15  20  15  6   1

      1
        7   21  35  35  21  7   1

 ** DONE **
```

# Pascal's Triangle—by Probability

This program simulates the dropping of balls through a triangular array shown below. At each level, a ball is equally likely to fall either to the left or right. At the bottom of the array a cup is placed at each end point; these cups collect the balls. After each group of balls has been dropped, the number of balls in each cup is tallied and displayed.



```
10 PRINT "PASCAL'S TRIANGLE
BY":"PROBABILITY":" "
20 INPUT "BALLS TO DROP =":M
30 INPUT "NUMBER OF LEVELS =
":K
35 PRINT " ":"POSITION","BAL
LS":" "
40 FOR N=1 TO M
50 T=0
60 FOR L=1 TO K
70 IF RND>.5 THEN 90
80 T=T+1
90 NEXT L
100 B(T+1)=B(T+1)+1
110 NEXT N
120 FOR L=1 TO K+1
130 PRINT L,B(L)
140 B(L)=0
150 NEXT L
```

```
PASCAL'S TRIANGLE BY
PROBABILITY

BALLS TO DROP =1000
NUMBER OF LEVELS =2

POSITION        BALLS

 1               234
 2               518
 3               248
```

```
>RUN
PASCAL'S TRIANGLE BY
PROBABILITY

BALLS TO DROP =1000
NUMBER OF LEVELS =4

POSITION          BALLS

 1                 51
 2                 229
 3                 407
 4                 240
 5                 73

** DONE **
```

The number of balls landing in the various cups at each level, when reduced to the lowest common denominator, should approximate the numbers in Pascal's Triangle. Do you know why they should?

Try a few runs of the program with a different number of balls. Do the numbers obtained really approximate those in Pascal's Triangle? If, at each dividing point in the array, every other ball went in the opposite direction, then Pascal's Triangle would be produced. Try running the program with 4 balls at Level 2; with 8 balls at Level 3; and with 16 balls at Level 4. Do your results look like those of the previous program which calculated the triangle? Does the approximation come closer as the number of balls is increased?

How can you determine how close your results from this program are to the exact value of Pascal's Triangle? One way is to take the number of balls that dropped into each cup on a given level and divide that by the total number of balls divided by the theoretical sum of the row. So, in the sample run at Level 4, you would divide the balls in each cup by 62.5 (1000/16). You then compare that to the "correct" number and compute the percent difference.

Here is the result of this procedure for Level 4 (actually the fifth row) in the sample run:

| *Cup* | *Balls* | *÷62.5* | *Correct* | *Deviation* |
|---|---|---|---|---|
| 1 | 51 | 0.82 | 1 | 18.00% |
| 2 | 229 | 3.66 | 4 | 8.50% |
| 3 | 407 | 6.51 | 6 | 8.50% |
| 4 | 240 | 3.84 | 4 | 4.00% |
| 5 | 73 | 1.17 | 1 | 16.80% |

Why are the outside cups "off" by a greater percentage than those closer to the center? Does this also happen on other levels with different numbers of balls?

# Common Birthdays

Here is an interesting little problem in probability. In a group of ten people selected at random, what is the probability that any of them will share the same birthday? How about a group of 20 people? Of 50 people?

Conversely, how many people would you need in a group such that there is a 50% probability that at least two of them have the same birthday? How many people would be needed for a 90% probability of an overlap?

Try to answer these questions, either by guess or by calculation, before you look at the output from the program.

This program provides a painless introduction to the world of statistics. The calculation is actually quite trivial. The probability that any person in a group has a birthday on January 1 is 1/365. If our group has only two people in it, the probability that both of them have a birthday on January 1 is 1/365 times 1/365, or a very small number indeed. The probability that they have a common birthday is 365 times the very small number just obtained, or about 0.27%.

However, if there are three people in the group, the probability goes up slightly. Call the three people Betsy, Ken, and Larry. From the reasoning above, we know that there is a 0.27% probability that Betsy and Ken have the same birthday; also a 0.27 % probability that Betsy and Larry share a birthday; and finally, a 0.27% chance that Larry and Ken have a common birthday. Hence, the total probability for a group of three people is three times the probability for just two people.

A group of four increases the probability over that of two people by a factor of six, five people by a factor of 10, six people by 15, seven people by 21, and so on. There is a progression here, but the probability can also be calculated by the formula:

$$P = 1 - \frac{365-N}{365}$$

Can you see why this formula produces the same result as the description in words and associated progression above?

Consider all the presidents of the United States. (How many have there been to date?) Two of them, James Polk and Warren Harding, were born on November 2. Is this to be expected given the size of the group?

Since birthdays can be predicted, at least statistically, it ought to be possible to predict deaths as well. It is interesting too that John Adams, James Monroe, and Thomas Jefferson all died on July 4th. Millard Fillmore and William Taft both died on March 8. What is the probability of that set of events?

```
10 PRINT TAB(11);"PROBABILIT
Y THAT"
20 PRINT "NO. OF       AT LEAS
T TWO HAVE"
30 PRINT "PEOPLE       THE SAM
E BIRTHDAY"
40 Q=364/365
50 FOR N=2 TO 40
60 P=100*(1-Q)
70 PRINT N;TAB(13);INT(P*100
+.5)/100;"%"
80 Q=Q*(365-N)/365
90 NEXT N
```

```
>RUN
               PROBABILITY THAT
NO. OF          AT LEAST TWO HAVE
PEOPLE          THE SAME BIRTHDAY
2                   .27 %
3                   .82 %
4                   1.64 %
5                   2.71 %
6                   4.05 %
7                   5.62 %
8                   7.43 %
9                   9.46 %
10                  11.69 %
11                  14.11 %
12                  16.7 %
13                  19.44 %

30                  70.63 %
31                  73.05 %
32                  75.33 %
33                  77.5 %
34                  79.53 %
35                  81.44 %
36                  83.22 %
37                  84.87 %
38                  86.41 %
39                  87.82 %
40                  89.12 %
```

Now it is your turn to write a program. Here is a game to be played among your friends. You make a bet that they have similar preferences in colors. Each person in the group puts up one penny and you, because you are so sure of yourself (and probability), put up an amount in cents equal to the number of people in the group. If you win, you get to keep all of the money; if your friends win, each of them gets double his original bet, or two cents.

Then each person selects a color (from a larger list than there are people, of course). If any two have picked the same color, you win; if all have picked different colors, they win.

Since you would like to win, you need to know how many colors should be on the list for different size groups to give you a better than 50-50 chance of winning. You might like the probability to be up around 70% or so. You can produce the necessary table with a five-line program. Go to it!

# Coins in a Pocket

The next two programs were originally written by Glenda Lappan and M.J. Winter and appeared in *Creative Computing* magazine. They are marvelous simulations for illustrating various aspects of probability.

Coins in a Pocket is a simulation of the following situation. A newspaper costs 5 cents. A customer has 5 pennies and a dime in his pocket and offers to pay for his paper by letting you, the vendor, select at random two of the six coins. If you and this customer repeat this procedure for the 20 working days of a month, how much more or less than $1.00 (20 days x 5 cents) are you likely to have collected?

The program below solves this little problem. The first random coin is selected from a group of six (Line 70 in which X = INT(RND*6)). The value of X can be 0, 1, 2, 3, 4, or 5. If it is 0, we assume the dime was selected and a second pick is not made, because it will surely be a penny. Thus 11 cents is added to the running total in Line 130.

If the first coin is a penny (X = 1, 2, 3, 4, or 5) then we make a second pick from the five remaining coins. Again, if the dime is chosen (Y = 0), 11 cents is added to the total; otherwise 2 cents is added.

As you can see from the sample runs, after a great number of trials, the average amount of money collected each day seems to be close to 5 cents. If you would like to convince yourself that the answer is, in fact, 5 cents, consider the following. Assign a letter to each coin, A - F. Let A be the dime and B through F be the pennies. Since all combinations are equally probable, here are all the possible combinations:

AB
AC BC
AD BD CD
AE BE CE DE
AF BF CF DF EF

There are five combinations with a dime (5 x 11 cents) and ten combinations of only pennies (10 x 2 cents). Add it up and you have 55 cents plus 20 cents divided by a total of 15 combinations which equals an average value of 75/15 or 5 cents.

```
10 RANDOMIZE
20 PRINT "PROGRAM SIMULATES TAKING TWO":"COINS AT RANDOM FROM A":"DIME AND 5 PENNIES"
30 PRINT
40 INPUT "NUMBER OF TRIALS = ":N
50 U=0
60 FOR K=1 TO N
70 X=INT(RND*6)
80 IF X=0 THEN 130
90 Y=INT(RND*6)
100 IF Y=0 THEN 130
110 U=U+2
120 GOTO 140
130 U=U+11
140 NEXT K
150 PRINT
160 PRINT "AVERAGE VALUE OF COINS IS":U/N;"CENTS"
```

```
PROGRAM SIMULATES TAKING TWO
COINS AT RANDOM FROM A
DIME AND 5 PENNIES

NUMBER OF TRIALS =1000

AVERAGE VALUE OF COINS IS
 4.934 CENTS

** DONE **

>RUN
PROGRAM SIMULATES TAKING TWO
COINS AT RANDOM FROM A
DIME AND 5 PENNIES

NUMBER OF TRIALS =100

AVERAGE VALUE OF COINS IS
 4.88 CENTS

** DONE **
```

# Baseball Cards

In statistics and probability, it is frequently necessary to predict the number of trials on average until one is successful. For example, if you are trying to roll a four with one die, on average how many tries will it take until you do so?

Since a die has six sides, the probability of rolling any number between 1 and 6 is $p = 1/6$. Thus, on any given roll, you have a $1/6$ chance of rolling a four and $5/6$ chance of rolling something else. This failure is designated q. So to be successful in rolling a four, we have a $1/6$ chance on the first roll. On two rolls, the probability is two trials x the probability of one failure, then success ($2 \times 5/6 \times 1/6$). On the third roll, the probability of success is three trials x the probability of two failures followed by one success ($3 \times 5/6 \times 5/6 \times 1/6$).

Continuing this reasoning leads to the formula for the expected number of trials to success:

$$E = 1p + 2q \times p + 3q \times p + 4q \times p + \ldots$$

This series can be reduced and solved for E which leads to:

$$E = \frac{1}{1-q} = \frac{1}{p}$$

So now the answer to the original problem can be calculated. The expected number of trials to roll a four is $1/p = 6$ tries until success.

But there is another way to approach this type of problem with the assistance of the computer. Consider the problem. Assume there are ten different prizes in Crunchies cereal boxes. How many boxes of cereal would you have to buy to obtain the complete set? The formula above can be expanded to solve this problem:

$$\frac{N}{N} + \frac{N}{N-1} + \frac{N}{N-2} + \cdots + \frac{N}{1} = N(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{N}$$

For a value of ten, you can solve this formula by hand. However, let's say you would like to solve this problem for baseball cards as found in packs of bubble gum. If there are 50 cards in a complete set, how many packs of gum must you buy, on average, to get a complete set. Write a computer program using the general formula above to solve for a set of any size. The answer for 50 cards is 224.96 packs of gum; how many would you have to buy for a set of 100 cards?

There is another way of approaching this problem. This method is similar to that used to randomly select coins out of the pocket. In this case, the random group is set equal to the total number of cards; this value is accepted as input in Statement 20. Each purchase is set equal to a random number between 1 and N (Statement 150). This card is then put into its proper place in the collection (Statement 160). However, if there is already a card there from

a previous purchase, we simply increment the purchase counter (Statement 190) but do not get any closer to obtaining a complete set. After each purchase, we test to see if the set is complete (Statement 200), otherwise we go on buying more packs of bubble gum.

When the set is complete, the number of packs of gum are tallied up and printed out. After a set of trials, the average is computed. This averaging value should be reasonably close to the value obtained by the formula although it will take a great number of trials before the two numbers are within 1% of each other.

```
10 PRINT "PROGRAM SIMULATES BUYING":"BUBBLE GUM WITH BASEBALL"
20 INPUT "CARDS. CARDS IN SERIES =":N
30 IF N<101 THEN 60
40 PRINT "SORRY, 100 IS MAXIMUM"
50 GOTO 20
60 INPUT "NUMBER OF SIMULATIONS =":K
70 DIM C(100)
80 PRINT " ":"TRIAL","PACKS"
90 S=0
100 FOR I=1 TO K
110 D=0
120 FOR J=1 TO N
130 C(J)=0
140 NEXT J
150 X=INT(RND*N)+1
160 C(X)=C(X)+1
170 IF C(X)=1 THEN 190
180 GOTO 150
190 D=D+1
200 IF D=N THEN 220
210 GOTO 150
220 T=0
230 FOR J=1 TO N
240 T=T+C(J)
250 NEXT J
260 PRINT I,T
270 S=S+T
280 NEXT I
290 PRINT "AVERAGE",S/K

>RUN
PROGRAM SIMULATES BUYING
BUBBLE GUM WITH BASEBALL
CARDS. CARDS IN SERIES =50
NUMBER OF SIMULATIONS =5

TRIAL          PACKS
 1             284
 2             246
 3             153
 4             223
 5             290
AVERAGE        239

** DONE **
```

```
>RUN
PROGRAM SIMULATES BUYING
BUBBLE GUM WITH BASEBALL
CARDS. CARDS IN SERIES =10
NUMBER OF SIMULATIONS =10

TRIAL              PACKS
  1                 23
  2                 14
  3                 26
  4                 27
  5                 19
  6                 28
  7                 25
  8                 20
  9                 26
  10                22
AVERAGE             23.2

** DONE **
```

If you run this second program for a set of 100 cards, it will sometimes run for a very long time before arriving at the answer. Try it with a set of 500 cards as are used by some real bubble gum manufacturers. You can go have your dinner while the program computes just the first value. Be sure to change the dimension statement in Line 70 to C(500).

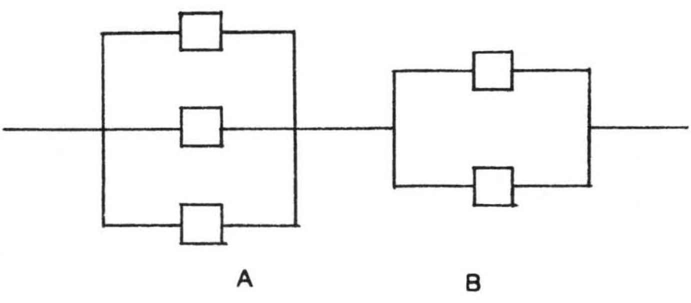
# System Reliability

As more and more people in the world come to depend upon mechanical and electronic devices in a myriad of different ways, it is important that these technological devices continue to function.

Some years ago, the military came up with a measure that could be applied to all kinds of systems, big and small, to measure reliability. It is called "mean time between failures." What this means is the length of time, on average, between breakdowns. For a tank, this may be 100 hours, while for a spacecraft, the MTBF must be considerably longer than the planned mission.

As we saw in an earlier section, it is frequently desirable to break down a large problem into smaller subproblems. To calculate the MTBF for a spacecraft would be quite impossible. Instead, it is necessary to start with smaller systems and build up to the whole.

Consider one of the electrical subsystems of a spacecraft. It uses five components as shown below, two parallel systems A and B, arranged in series. The parallel subsystems are said to be "redundant." This is one method of increasing reliability since the system will continue to work if at least one of the components works.




If the manufacturer of the components stated that each one has a 60% probability of lasting 1000 hours, what is the chance that the entire system will last 1000 hours? Take a guess before reading on to the solution below. Is your guess greater than 60% or less? Why?

The program below is a simulation of this system. Remember in subsystem A, it will continue to work if any of the three components works and in B if either of the two components works. However, the system will fail if all three of the A components or both of the B components fail.

In Statement 40, the program is set up to make 500 trials of the system. Statement 60 selects a random integer between 1 and 10 for each component for each trial. If the value of this integer is 6 or under, the component is okay (60% probability of working at the end of 100 hours). If it is 7, 8, 9, or 10, this means the component has failed, and the program prints "Bad!"

```
10 RANDOMIZE
20 PRINT "PROGRAM SIMULATES
USE OF";"CIRCUIT FOR 1000 HO
URS":""
30 S=0
40 FOR I=1 TO 500
50 FOR J=1 TO 5
60 C(J)=INT(RND*10)+1
70 IF C(J)>6 THEN 100
80 C(J)=1
90 GOTO 110
100 C(J)=0
110 NEXT J
120 IF C(1)=1 THEN 170
130 IF C(2)=1 THEN 170
140 IF C(3)=1 THEN 170
150 PRINT "BAD!";
160 GOTO 220
170 IF C(4)=1 THEN 210
180 IF C(5)=1 THEN 210
190 PRINT "BAD!";
200 GOTO 220
210 S=S+1
220 NEXT I
230 PRINT "RELIABILITY ABOUT
";INT(10000*(S/500)+.5)/100;
"%"
999 END
>

PROGRAM SIMULATES USE OF
CIRCUIT FOR 1000 HOURS

BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!BAD!
BAD!BAD!BAD!BAD!BAD!BAD!
RELIABILITY ABOUT 77.8 %

** DONE **
```

Statements 120-140 tests whether subsystem A works and Statements 170-180 checks out subsystem B. If both work, a success is counted in Statement 210.

Now it is your turn to use what you have learned in this chapter and write a program to determine the mean time between failures of the system above.

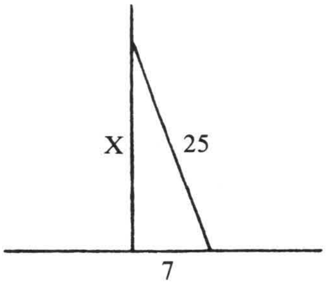
# 7

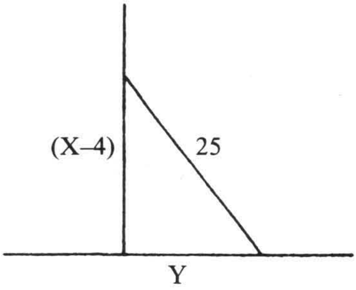
# Geometry and Calculus

In this chapter several of the problem solving approaches from earlier chapters are used to solve geometric problems. You will find that many problems can be solved in a variety of different ways—by applying a formula, by trial and error, by successive approximations, and, in some cases, by common sense. Perhaps the most important thing to learn from these problems and programs is how to analyze a problem to reach the solution quickly and painlessly.

# Crossed and Slipping Ladders

Here is a simple problem of a slipping ladder. A ladder 25 feet long is placed so its foot is 7 feet from the base of a building. The base of the ladder slipped on some loose gravel so that the top is 4 feet lower than where it was to start. How far did the foot of the ladder slip?




The diagrams show the two positions of the ladder. By the Pythagorean theorem we know that $a^2 + b^2 = c^2$, hence, the equations needed to solve the problem are:

$$x = \sqrt{25^2 - 7^2}$$

$$y = \sqrt{25^2 - (x\text{-}4)^2}$$

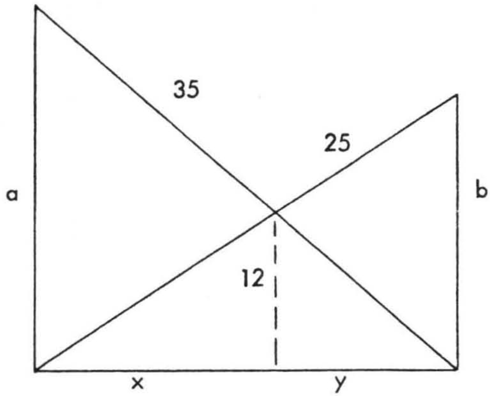
and the amount of slippage of the base is $z = y - 7$. These three equations are put into a computer program which quickly calculates an answer of 8 feet.

While the aritmetic in the above problem is not particularly messy, it is still no great joy to solve by hand. However, the computer is just as happy to do the problem with really messy dimensions, say a ladder length of 27.83 feet and a distance from the wall of 7.62 feet.

```
10 PRINT "SLIPPING LADDER"
20 C=25
30 B=7
40 X=SQR(C^2-B^2)
50 Y=SQR(C^2-(X-4)^2)
60 Z=Y-B
70 PRINT "BASE SLIPPED";Z;"F
EET"

>RUN
SLIPPING LADDER
BASE SLIPPED 8 FEET

** DONE **
```

```
>20 C=27.83
>30 B=7.62

>RUN
 SLIPPING LADDER
 BASE SLIPPED 8.386131551
 FEET

 ** DONE **
```

Let's consider an old problem found in many classic collections. Two ladders, one 25 feet long and the other 35 feet long lean against buildings on opposite sides of an alley as shown below. The point at which the ladders cross is 12 feet above the ground. How wide is the alley?




By using similar triangles twice we find that

$$12/a + 12/b = 1$$

Then, by applying the Pythagorean theorem and reducing, we obtain:

$$a^2 - b^2 = 600$$

Using one of the methods described in the Problem Solving chapter, you can solve these two simultaneous equations. Or they can be combined into one equation in which z, the width of the alley is

$$z = \sqrt{35^2 - a^2}$$

By eliminating b, the following fouth degree equation is obtained

$$a^4 - 24a^3 - 600a^2 + 14400a - 86400 = 0$$

Again, this can be solved using one of the methods in the chapter on Problem Solving. This is the traditional approach, but there is another. The solution is the intersection of the curves described by the two original equations. By using successive approximations, say in steps of 1 for a, you could solve for $b_1$ and $b_2$ in the following restated original equations

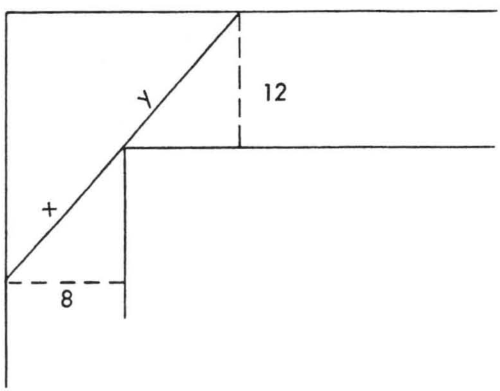
$$b_1 = \frac{12a}{a - 12}$$

$$b_2 = \sqrt{600 - a^2}$$

When $b_1$ falls below $b_2$, reduce the step to 0.1 to obtain a closer approximation. By continuing this method of successive approximations, it is possible to obtain a very accurate solution.

Is there another approach? Yes, there is and it also avoids the quartic equation. It uses the original equations in a trial and error procedure as described in the chapter on Sets and Repetitive Trials. See if you can write a program using this approach.

Here is another classic problem for you to solve in any way that you like. What is the longest ladder that can be carried in a horizontal position around the corner made where a 12-foot wide alley meets one that is 8 feet wide. The diagram shows the problem.



# Distance Between Coordinate Points

This program solves for the distance between any two points in three-dimensional space defined by their x, y, and z coordinates.

The formula for the distance between two points in three-dimensional space is:

$$d = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2$$

It would be desirable for the program to be able to solve any problem of this kind. One approach would be to use INPUT statements to accept the point coordinates. Another is to use a DATA statement to define the points. Then only this one statement has to be changed for a new problem or set of problems.

The program is written to calculate the distance between three sets of points. The points used were:

| | | |
|---|---|---|
| 0, 0, 0 | and | 3, 4, 5 |
| 3.5, -4.7, 6.2 | and | - 0.9, 3.0, 4.4 |
| 67, 36, 82 | and | 54, 25, 90 |

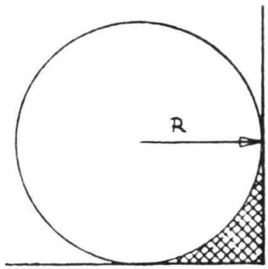
This calculation is used extensively in aerospace navigation. Can you determine the angle or "compass heading" of the resulting flight path? Better start with just two dimensions.

```
10 PRINT "PROGRAM CALCULATES
 DISTANCE":"BETWEEN POINTS I
N THREE-":"DIMENSIONAL SPACE
":"
20 PRINT "     POINT"
30 PRINT "COORDINATES      DIS
TANCE":"     "
40 READ A,B,C,D,E,F
50 L=SQR((A-D)^2+(B-E)^2+(C-
F)^2)
60 PRINT A;B;C
70 PRINT D;E;F;TAB(16);L
80 PRINT
90 GOTO 40
100 DATA 0,0,0,3,4,5
110 DATA 3.5,-4.7,6.2,-.9,3,
4.4
120 DATA 67,36,82,54,25,90
>
>
>RUN
PROGRAM CALCULATES DISTANCE
BETWEEN POINTS IN THREE-
DIMENSIONAL SPACE

     POINT
COORDINATES      DISTANCE

 0   0   0
 3   4   5          7.071067812

 3.5 -4.7 6.2
-.9  3   4.4        9.049309366

 67  36  82
 54  25  90         18.81488772

 * DATA ERROR IN 40
```

# Area—by Calculation

It is a simple matter to calculate the area of regular geometric figures by using the usual formulae. However, the problem becomes more difficult when it is necessary to calculate the area of two combined regular shapes or the area of irregular shapes. This program shows the method of analysis for solving a problem of the first type, while the next program demonstrates four methods of dealing with irregular areas.

The problem is to solve for the shaded area of the figure below for any value of the radius, R.




The first step is to recall the formulae for calculating the area of a circle and square:

$$A \text{ (circle)} = pi * R^2$$

$$A \text{ (square)} = Side^2 \text{ or } (2 * R)^2$$

The difference in area between a square and a circle inscribed within its borders is:

$$A \text{ (difference)} = A \text{ (square)} - A \text{ (circle)}$$

and the area of one corner is the difference divided by 4. The program below will calculate the area for any radius.

Now it is your turn. Write a program to calculate the difference in area between a circle and square in which the square is inscribed within the circle. Now, change your program to calculate the difference in area for a triangle, a hexagon and an octagon inscribed within a circle.

```
10 PRINT "PROGRAM CALCULATES
 AREA":"BETWEEN A CIRCLE INS
CRIBED":"WITHIN A SQUARE":"
"
20 INPUT "RADIUS =":R
30 C=3.14159*R^2
40 S=(2*R)^2
50 D=S-C
60 D1=D/4
70 PRINT "TRAPPED AREA =";D
80 PRINT "ONE CORNER =";D1
```

```
>RUN
PROGRAM CALCULATES AREA
BETWEEN A CIRCLE INSCRIBED
WITHIN A SQUARE

RADIUS =10
TRAPPED AREA = 85.841
ONE CORNER = 21.46025

** DONE **

>RUN
PROGRAM CALCULATES AREA
BETWEEN A CIRCLE INSCRIBED
WITHIN A SQUARE

RADIUS =3.14159
TRAPPED AREA = 8.472152802
ONE CORNER = 2.1180382

** DONE **
```

Extend your program to calculate the difference in area for any regular figure inscribed within a circle of radius 1.0. Set up a table or results as follows:

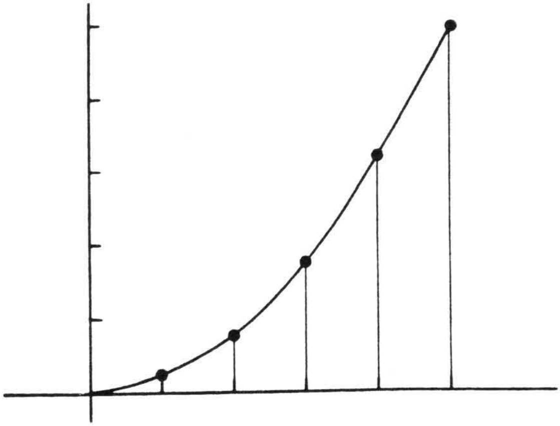
| *Number of sides* | *Difference in Area* |
|---|---|
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |

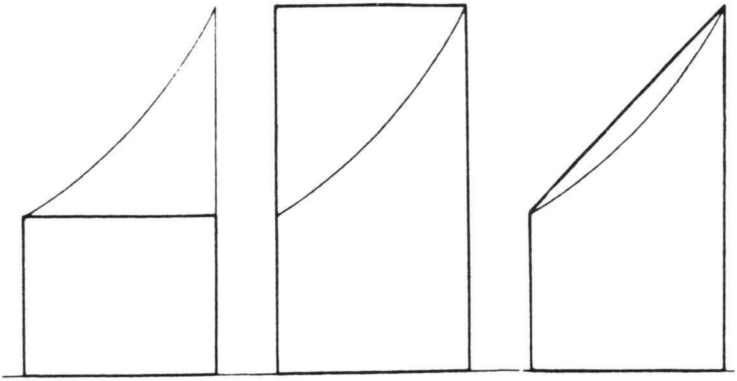
What can you conclude from these results? Do these areas follow in some sort of progression?

# Area—by Integration

In many cases it is necessary to determine the area of an irregular figure or the area under a curve where an exact formula is not available. The approach most commonly used is to divide up the enclosed area into small regularly shaped pieces and sum up the areas of all of these pieces.

The easiest shape to use in these calculations is a rectangle. A group of rectangles can either be inscribed within the irregular figure or curve, or circumscribed around it. The first two diagrams show these two methods being used to find the area under a curve. A third method is to use trapezoids which, depending upon the direction of curvature, will either be inscribed or circumscribed automatically.





Inscribed Rectangle　　Circumscribed Rectangle　　Trapezoid

A fourth method, known as Simpson's Rule, essentially fits a series of parabolas between the points of the curve and calculates the average area. It requires that the area be divided into an even number of parallel slits. Let us call the total number of divisions 2m which are h distance apart. The point on the curve where the first line intersects is $y_0$, the second $y_1$, and so on until $y_{2m}$. The area is then given by the formula:

$$A = \tfrac{1}{3}h[(y_0 + y_{2m}) + 4(y_1 + y_3 + \ldots + y_{2m-1}) + 2(y_2 + y_4 + \ldots y_{2m-2})]$$

The area as calculated by Simpson's Rule converges extremely quickly compared to the other methods. Nevertheless, as the number of intervals increases, all the methods approach the same limit. As might be expected, the trapezoidal approach converges more quickly than either of the approaches using rectangles. Compute the average of the two methods using rectangles. What do you get? Does this suggest another method?

The methods used in this program involve the calculus. And you thought the calculus was difficult! Now you know otherwise.

```
10 PRINT "PROGRAM COMPUTES A
REA UNDER":"A CURVE BY FOUR
METHODS OF":"NUMERICAL INTEG
RATION":""
20 PRINT "DEFINE CURVE IN LI
NE 100"
30 PRINT "INPUT START AND EN
D VALUES"
40 INPUT "FOR X (SMALLER FIR
ST)=":A,B
50 PRINT
60 PRINT "NO. INTERVALS":"IN
SCR RECTS    TRAPEZOIDS":"CI
RCUM RECTS   PARABOLAS"
70 PRINT
80 M=-2
90 S=0
100 DEF FNA(X)=X^3
110 M=M+3
120 FOR N=M TO M+2
130 C=S
140 Q=0
150 P=0
160 D=2^N
170 H=(B-A)/D
180 FOR I=0 TO D-1
190 X=A+I*H
200 P=P+H*FNA(X)
210 Q=Q+H*FNA(X+H)
220 NEXT I
```

```
230 T=(P+Q)/2
240 U=FNA(A)+FNA(B)
250 FOR J=2 TO (D-2)STEP 2
260 U=U+2*FNA(A+J*H)
270 NEXT J
280 V=0
290 FOR K=1 TO (D-1)STEP 2
300 V=V+4*FNA(A+K*H)
310 NEXT K
320 S=(U+V)*(H/3)
330 PRINT D:P,T:Q,S
340 NEXT N
350 IF D<64 THEN 110
360 IF ABS((C-S)/((C+S)/2))>
.0001 THEN 110
370 END
```

```
>RUN
PROGRAM COMPUTES AREA UNDER
A CURVE BY FOUR METHODS OF
NUMERICAL INTEGRATION

DEFINE CURVE IN LINE 100
INPUT START AND END VALUES
FOR X (SMALLER FIRST)=1,10

NO. INTERVALS
INSCR RECTS      TRAPEZOIDS
CIRCUM RECTS     PARABOLAS

 2
 753.1875        3000.9375
 5248.6875       2499.75
 4
 1501.171875     2625.046875
 3748.921875     2499.75
 8
 1969.136719     2531.074219
 3093.011719     2499.75
 16
 2226.612305     2507.581055
 2788.549805     2499.75
 32
 2361.223389     2501.707764
 2642.192139     2499.75
 64
 2429.997253     2500.239441
 2570.481628     2499.75

** DONE **
```

*"According to the computer simulation, it should hit the earth in 0.00298 seconds."*

# 8

# Science

Using techniques and approaches presented in the previous chapters, this chapter contains five programs in the area of science. One uses a formula to solve simple gas problems, two are drill exercises on the gas laws of Boyle and Charles, and the last two are simulations. You will see that the simulations draw upon many previous techniques such as progressions and repetitive calculations.

# Gas Volume

Here is a simple program to produce a not-so-simple table of values for gas volumes.

The volume of a gas varies directly with the absolute temperature T (Kelvin) and inversely with the pressure P. If a certain quantity of gas occupies 500 cubic feet at a pressure of 53 pounds per square foot and an absolute temperature of 500 degrees, what volume will it occupy at 600 degrees absolute temperature and pressures from 100 to 1000 pounds per square foot in increments of 50 pounds?

The original conditions are used to solve for the constant K in Statement 40 (K = V*P/T). Then new volumes are computed for varying pressures with T equal to 600 degrees. The formula used is V = K*T/P.

In the second part of the program, Lines 70 to 100 are replaced to produce a plot of the gas volume for the various pressures.

How would you modify the program to deal with a more general case (i.e., other gasses and different temperatures)? Second, can you write a program that produces a table of values for a gas at different pressures and temperatures?

```
10 V=500
20 P=53
30 T=500
40 K=V*P/T
50 PRINT "PROGRAM COMPUTES G
AS VOLUMES":"AT DIFFERENT PR
ESSURES":" "
60 INPUT "TEMPERATURE (K) ="
:T1
70 PRINT " ":"PRESSURE","VOL
UME"
80 FOR P=100 TO 1000 STEP 50
90 V=K*T1/P
100 PRINT P,V
110 NEXT P
```

```
>RUN
PROGRAM COMPUTES GAS VOLUMES
AT DIFFERENT PRESSURES

TEMPERATURE (K) =600

PRESSURE          VOLUME
 100              318
 150              212
 200              159
 250              127.2
 300              106
 350              90.85714286
 400              79.5
```

```
 450                     70.66666667
 500                     63.6
 550                     57.81818182
 600                     53
 650                     48.92307692
 700                     45.42857143
 750                     42.4
 800                     39.75
 850                     37.41176471
 900                     35.33333333
 950                     33.47368421
 1000                    31.8

** DONE **
```

```
10 V=500
20 P=53
30 T=500
40 K=V*P/T
50 PRINT "PROGRAM COMPUTES GAS VOLUMES":"AT DIFFERENT PRESSURES":
60 INPUT "TEMPERATURE (K) =":T1
70 PRINT "PRES/100      PLOT OF VOLUME"
80 FOR P=100 TO 1000 STEP 50
90 V=K*T1/P
100 PRINT P/100;TAB(4+INT(V/13));"*"
110 NEXT P
```

```
>RUN
PROGRAM COMPUTES GAS VOLUMES
AT DIFFERENT PRESSURES

TEMPERATURE (K) =600

PRES/100      PLOT OF VOLUME
 1                                *
 1.5                      *
 2                    *
 2.5                *
 3                 *
 3.5            *
 4              *
 4.5           *
 5            *
 5.5          *
 6            *
 6.5         *
 7           *
 7.5         *
 8           *
 8.5        *
 9          *
 9.5        *
 10         *
```

# Charles' Law Drill

In the previous program, gas volumes were calculated using Charles' Law and Boyle's Law. Here is a drill and practice program (remember Chapter 1?) that produces problems relating the volume and temperature of a gas. When pressure is constant, the volume/temperature relationships can be stated as follows:

$$\frac{V_0}{T_0} = \frac{V_1}{T_1}$$

The program presents four problems, one to solve for each of the four variables in the equation above.

How can you make the program more efficient? More interesting?

```
10 RANDOMIZE
20 PRINT "DRILL ON CHARLES'
LAW":""
30 PRINT "VOLUMES IN MILLILI
TRES":"TEMPERATURES IN DEGRE
ES K":""
40 V=INT(RND*50+50)*100
50 V1=INT(RND*50+50)*100
60 T=INT(RND*125+125)
70 T1=INT(RND*125+125)
80 C=C+1
90 ON C GOTO 100,140,180,220
,260
100 V=0
110 PRINT "SOLVE FOR V WHEN"
120 Q1=V1*T/T1
130 GOTO 270
140 V1=0
150 PRINT "WHAT IS V1 GIVEN"
160 Q1=V*T1/T
170 GOTO 270
180 T=0
190 PRINT "CALCULATE THE VAL
UE OF T"
200 Q1=V*T1/V1
210 GOTO 270
220 T1=0
230 PRINT "SOLVE FOR T1 GIVE
N"
240 Q1=T*V1/V
250 GOTO 270
260 STOP
270 PRINT " V      V1      T
   T1"
280 PRINT V;TAB(7);V1;TAB(14
);T;TAB(21);T1
290 INPUT "YOUR ANSWER =":Q
300 PRINT "CORRECT VALUE =";
Q1
310 PRINT
320 GOTO 40
```

```
>RUN
DRILL ON CHARLES' LAW

VOLUMES IN MILLILITRES
TEMPERATURES IN DEGREES K

SOLVE FOR V WHEN
 V         V1        T         T1
 0         9800      169       240
YOUR ANSWER =6600
CORRECT VALUE = 6900.833333

WHAT IS V1 GIVEN
 V         V1        T         T1
 6800      0         152       146
YOUR ANSWER =6500
CORRECT VALUE = 6531.578947

CALCULATE THE VALUE OF T
 V         V1        T         T1
 6800      8900      0         208
YOUR ANSWER =161
CORRECT VALUE = 158.9213483

SOLVE FOR T1 GIVEN
 V         V1        T         T1
 8600      6800      248       0
YOUR ANSWER =200
CORRECT VALUE = 196.0930233

** DONE **
```

# Boyle's Law Drill

Boyle's Law describes the behavior of gases under ideal conditions when pressure and volume are varied. When the temperature is constant, Boyle found that:

$$P_0 * V_0 = P_1 * V_1$$

Pressure is normally measured in centimeters of mercury while volume is measured in millilitres. As with the drill on Charles' Law, this program presents four problems, one to solve for each of the four variables in the equation above.

Unlike many other drill and practice programs, these two do not compare your answer with the correct one. Instead, they leave that up to you to do. Is this desirable? Why or why not? If you feel it is undesirable, change the programs so they do compare answers and calculate a score.

```
10 RANDOMIZE
20 PRINT "DRILL ON BOYLE'S L
AW":" "
30 PRINT "VOLUMES IN MILLILI
TRES":"PRESSURES IN CM OF ME
RCURY":" "
40 V=INT(RND*50+50)*100
50 V1=INT(RND*50+50)*100
60 P=INT(RND*150+150)
70 P1=INT(RND*150+150)
80 C=C+1
90 ON C GOTO 100,140,180,220
,260
100 V=0
110 PRINT "SOLVE FOR V WHEN"
120 Q1=P1*V1/P
130 GOTO 270
140 V1=0
150 PRINT "WHAT IS V1 GIVEN"
160 Q1=P*V/P1
170 GOTO 270
180 P=0
190 PRINT "CALCULATE THE VAL
UE OF T"
200 Q1=P1*V1/V
210 GOTO 270
220 P1=0
230 PRINT "SOLVE FOR T1 GIVE
N"
240 Q1=P*V/V1
250 GOTO 270
260 STOP
270 PRINT " V      V1      P
    P1"
280 PRINT V;TAB(7);V1;TAB(14
);P;TAB(21);P1
290 INPUT "YOUR ANSWER =":Q
300 PRINT "CORRECT VALUE =";
Q1
310 PRINT
320 GOTO 40
```

```
>RUN
DRILL ON BOYLE'S LAW

VOLUMES IN MILLILITRES
PRESSURES IN CM OF MERCURY

SOLVE FOR V WHEN
 V        V1       P         P1
 0        7400     251       249
YOUR ANSWER =7350
CORRECT VALUE = 7341.035857

WHAT IS V1 GIVEN
 V        V1       P         P1
 9400     0        232       215
YOUR ANSWER =10060
CORRECT VALUE = 10143.25581

CALCULATE THE VALUE OF T
 V        V1       P         P1
 8800     6700     0         193
YOUR ANSWER =144
CORRECT VALUE = 146.9431818

SOLVE FOR T1 GIVEN
 V        V1       P         P1
 6300     9700     265       0
YOUR ANSWER =170
CORRECT VALUE = 172.1134021

** DONE **
```

# Photoelectric Emissions

When light of a short wavelength falls on a metal surface, electrons are emitted from the metal. According to the description of this phenomenon by Einstein, there is a maximum wavelength for every metal above which no electrons are emitted. This is called the critical wavelength of the metal.

This program simulates a laboratory experiment in which a metal is placed in a vacuum and bombarded with soft X-rays. The number of electrons emitted is collected and measured with a microammeter. The program simulates three trials at each of nine wave lenghts.

After each set of experimental data, the program asks if you would like another run at a higher light intensity. The reason for doing this is that sometimes at low light intensities, not enough electrons have been emitted for meaningful measurements.

You can increase the precision of the experiment by increasing the number of wavelengths at which it is run. This value can be changed in Statement 60; note that the variable L is divided into 1000 in order to express the wavelength in Angstroms. One Angstrom (A) equals $10^{-8}$ centimeters or $10^{-4}$ microns.

Here are the coefficients for several metals:

| | |
|---|---|
| Silver | .308 |
| Bismuth | .338 |
| Cadmium | .318 |
| Lead | .340 |
| Platinum | .385 |

It is a rare physics laboratory in a high school or college today that has the experimental apparatus to run this experiment, yet with a small computer the equipment can be simulated. There are many other things with which you would not normally be able to experiment that can be simulated with a computer. Things such as a nuclear power plant, a malaria epidemic, an urban mass transit system, and a bicycle factory.

Can you write a simulation for a real world system? You will find sections on simulations in the books *Computers in Mathematics* and *Computers in Science and Social Studies.* These, along with articles in *99'er* and *Creative Computing* magazines, might be of some help in writing a simulation of your own.

```
10 PRINT "PROGRAM SIMULATES
METAL":"BOMBARDMENT WITH SOF
T X-RAYS":" "
20 INPUT "COEFFICIENT OF MET
AL =":V
30 K=INT(1+2*RND)
40 PRINT "OUTPUT IN MICROAMP
ERES":" "
50 PRINT "WAVE  TRIAL  TRIAL
  TRIAL":"LENGTH   1      2
    3":" "
60 FOR L=.42 TO .25 STEP -.0
2
70 M=INT(1000/L)
80 PRINT M;
90 FOR J=1 TO 3
100 IF L>V THEN 130
110 I=SQR(INT(25*RND))
120 GOTO 140
130 I=SQR(K*K*100+INT(35*RND
))
140 N=INT(10*I+.5)/10
150 PRINT TAB(J*7);N;
160 NEXT J
170 PRINT
180 NEXT L
190 PRINT " ":"INCREASE LIGH
T INTENSITY":"(Y OR N)";
200 INPUT A$
210 IF A$="N" THEN 270
220 PRINT "BY WHAT FACTOR (1
 TO 10)";
230 INPUT F
240 K=K*F
250 PRINT
260 GOTO 50
270 STOP
```

```
>RUN
 PROGRAM SIMULATES METAL
 BOMBARDMENT WITH SOFT X-RAYS

 COEFFICIENT OF METAL =.34
 OUTPUT IN MICROAMPERES

 WAVE    TRIAL   TRIAL   TRIAL
 LENGTH    1       2       3

  2380   20.3    20.4    20.3
  2500   20.2    20.5    20.2
  2631   20.5    20.6    20
  2777   20.8    20.4    20.4
  2941   2.8     1.7     4.5
  3125   4.1     1.7     4.4
  3333   3       4.4     4.6
  3571   4.1     3.7     1
  3846   3       4       3.5

 INCREASE LIGHT INTENSITY
 (Y OR N)? Y
 BY WHAT FACTOR (1 TO 10)? 10

 WAVE    TRIAL   TRIAL   TRIAL
 LENGTH    1       2       3

  2380   200     200.1   200.1
  2500   200.1   200.1   200
  2631   200.1   200.1   200
  2777   200.1   200.1   200
  2941   4.6     3.6     2.2
  3125   4.4     3.5     3.9
  3333   2.4     3.3     4.4
  3571   3.7     3.7     2.4
  3846   3       4.5     3.7

 INCREASE LIGHT INTENSITY
 (Y OR N)? N

 ** DONE **
```

# Mutation of Moths

This program is a simulation of the growth of a colony of pepper moths. The program allows a genetic mutation to be introduced in some year between 1 and 30. The mutation can favor either dark or light colored moths.

The program as it appears starts with a total colony of light moths; you could add an initial group of dark colored moths in Statement 35 as P1.

The sample run shows a mutation which favors dark moths occurring in Year 3. The mutation affects about 2% of the light moths each year and causes them to become dark. The program displays the number of moths of each color over a 30-year period.

Obviously, because of the long time period involved it would be difficult to carry out this experiment in school, so the computer again is of real benefit.

```
10 PRINT "PROGRAM SIMULATES
MUTATION":"OF A COLONY OF MO
THS":""
20 INPUT "RATE OF MUTATION (
1-10)=":M
30 P0=10000
40 Z=P0
50 PRINT "IN WHAT YEAR DOES
CHANGE":"IN ENVIRONMENT OCCU
R";
60 INPUT X
70 L=1
80 D=2
90 PRINT "CHANGE FAVOR LIGHT
 OR DARK":"MOTHS (L OR D)";
100 INPUT E$
110 PRINT " ":"YEAR DARK MOT
HS  LIGHT MOTHS"
120 FOR T=1 TO 30
130 IF T>=X THEN 160
140 P1=0
150 GOTO 220
160 IF E$<>"D" THEN 140
170 P1=INT(P1+.01*M*P0+.5)
180 P0=INT(Z-P1+.5)
190 IF P1<Z THEN 220
200 P1=Z
210 P0=0
220 PRINT T;TAB(8);P1;TAB(19
);P0
230 NEXT T
```

```
>RUN
PROGRAM SIMULATES MUTATION
OF A COLONY OF MOTHS

RATE OF MUTATION (1-10)=2
IN WHAT YEAR DOES CHANGE
IN ENVIRONMENT OCCUR? 3
CHANGE FAVOR LIGHT OR DARK
MOTHS (L OR D)? D

YEAR DARK MOTHS  LIGHT MOTHS
1     0           10000
2     0           10000
3     200         9800
4     396         9604
5     588         9412
6     776         9224
7     960         9040
8     1141        8859
9     1318        8682
10    1492        8508
11    1662        8338
12    1829        8171
13    1992        8008
14    2152        7848
15    2309        7691
16    2463        7537
17    2614        7386
18    2762        7238
19    2907        7093
20    3049        6951
21    3188        6812
22    3324        6676
23    3458        6542
24    3589        6411
25    3717        6283
26    3843        6157
27    3966        6034
28    4087        5913
29    4205        5795
30    4321        5679
```

The so-called "killer bees" that were introduced into South America in the mid 70's were supposed to help honey production because they were much more energetic than the lazy honey bees in Brazil. The idea was that they would mate with the existing honey bees and produce a more productive strain. However, they went on a rampage killing people and terrorizing the country. Then they started to migrate north. U.S. agricultural officials became alarmed that they would invade this country and bring death and destruction.

Over the years, the killer bees have started to mate with the more docile South American honey bees, but only at the rate of 3% per year. Assuming they pose a danger to the U.S. until their numbers are reduced to 15% of their original quantity, how many years will it take for this to occur? The present migration patterns will bring them to the U.S. by 1989; will they still be dangerous (assume that the year of introduction was 1974)?

## Projectile Motion

The path followed by a projectile is called its trajectory. The trajectory is affected to a large extent by air resistance, which makes an exact analysis of the motion extremely complex. We shall, however, neglect the effects of air resistance and assume the motion takes place in empty space.

In the general case of projectile motion, the body (bullet, rocket, mortar, etc.) is given an initial velocity at some angle $\theta$ above (or below) the horizontal.

If $V_o$ represents the initial velocity (muzzle velocity), the horizontal and vertical components are:

$$V_{ox} = V_o \cos\theta \text{ , } V_{oy} = V_o \sin\theta$$

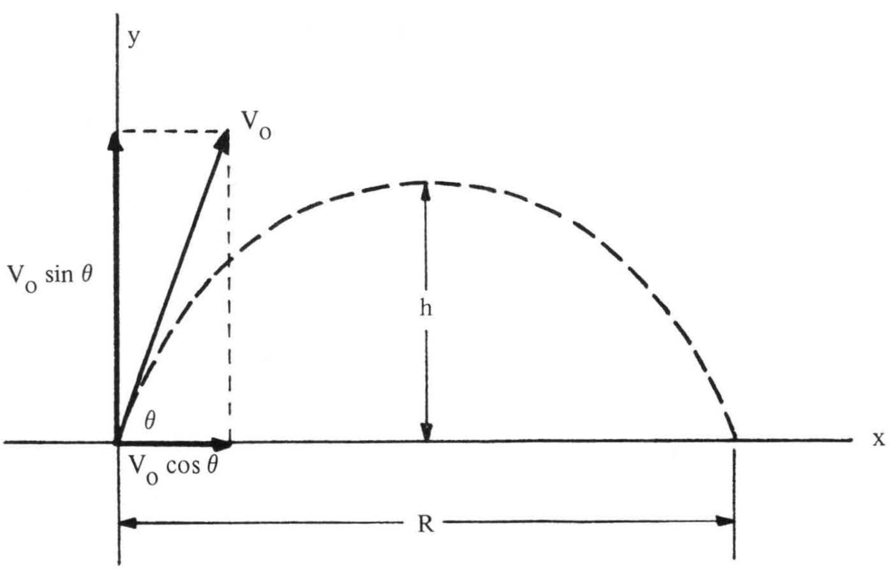
Since we are neglecting air resistance, the horizontal velocity component remains constant throughout the motion. At any time, it is:

$$V_x = V_{ox} = V_o \cos\theta = \text{constant} \quad (1)$$

The vertical part of the motion is one of constant downward acceleration due to gravity. It is the same as for a body projected straight upward with an initial velocity $V_o \sin \theta$. At a time "t" after the start, the vertical velocity is:

$$V_y = V_{oy} - gt = V_o \sin\theta - gt \quad (2)$$

where "g" is the acceleration due to gravity.



The horizontal distance is given by:

$$x = V_{ox}t = (V_o \cos\theta)\, t \qquad (3)$$

and the vertical distance by:

$$\begin{aligned} y &= V_{oy}t - 1/2\, gt^2 \\ &= (V_o \sin\theta)\, t - 1/2\, gt^2 \end{aligned} \qquad (4)$$

The time for the projectile to return to its initial elevation is found from Equation (4) by setting $y = 0$. This gives

$$t = \frac{2\, V_o \sin\theta}{g} \qquad (5)$$

The horizontal distance when the projectile returns to its initial elevation is called the horizontal range. "R." Introducing the time to reach the point in Equation (3), we find:

$$R = \frac{2\, V_o^{\,2} \sin\theta \cos\theta}{g} \qquad (6)$$

Since $2\sin\theta\cos\theta = \sin 2\theta$, Equation 6 becomes:

$$R = \frac{V_o^{\,2} \sin 2\theta}{g} \qquad (7)$$

The horizontal range is thus proportional to the square of the initial velocity for a given angle of elevation. Since the maximum value of $\sin 2\theta$ is 1, the maximum horizontal range, Rmax is $V_O^2/g$. But if $\sin 2\theta = 1$, then $2\theta = 90°$ and $\theta = 45°$. Hence the maximum horizontal range, in the absence of air resistance, is attained with angle of elevation of 45°.

From the standpoint of gunnery, what one usually wishes to know is what the angle of elevation should be for a given muzzle velocity $v_O$ in order to hit a target whose position is known. Assuming the target and gun are at the same elevation and the target is at a distance R, Equation (7) may be solved for $\theta$.

$$\begin{aligned} \theta &= 1/2 \sin^{-1}\left(\frac{Rg}{V_o^{\,2}}\right) \\ &= 1/2 \sin^{-1}\left(\frac{R}{R_{max}}\right) \end{aligned} \qquad (8)$$

Provided R is less than the maximum range, this equation has two solutions for values of $\theta$ between 0° and 90°. Either of the angles gives the same range. Of course, time of flight and maximum height reached are both greater for the high angle trajectory.

For example, say the maximum range of our gun is 10,000 yards and the target is at 5,900 yards:

$$\theta = 1/2 \sin^{-1} \frac{5,900}{10,000}$$

$$= 1/2\ 36^\circ$$

$$= 18^\circ, \text{ or } 90^\circ - 18^\circ = 72^\circ$$

Try the computer game Gunner which appears below. Use trial and error to try and destroy the target (see sample run). You get five chances per target. How many shots did it take to destroy all five targets? Did you ever fail to destroy a target in five trials?

From the discussion above, you should realize that 45 degrees of elevation provides maximum range with values over or under 45° providing less range.

The maximum range of the gun will vary between 20,000 and 60,000 yards and the burst radius of a shell is 100 yards.

You can also determine the correct firing angle from sine tables, or a slide rule, a scientific calculator or a computer. You should be able to destroy every target with just one shot. What happens when the target is very close? Can you always use whole angles?

Now write a computer program to accept the maximum range of your gun and the range to the target and then calculate the correct firing angle. You will have to solve two problems to write such a program:

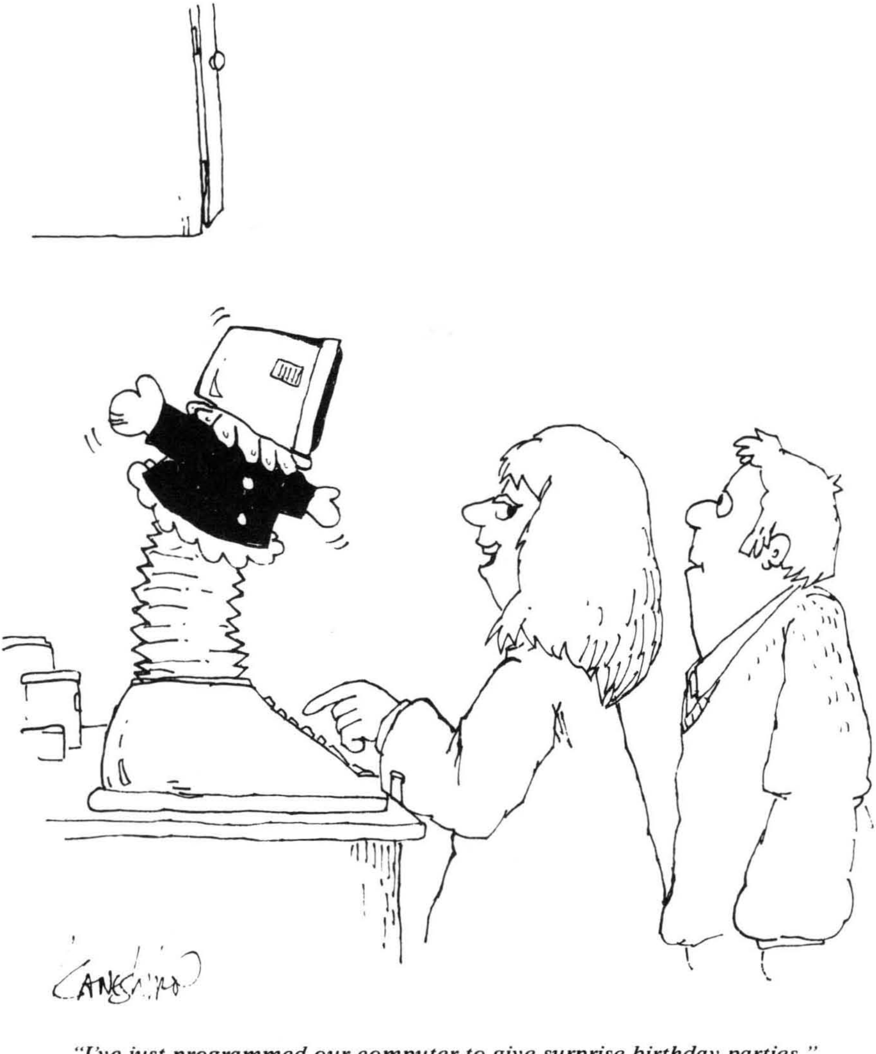
1. The Basic language does not have an ARCSIN function. However, the following formula may help.

$$\sin^{-1} x = \tan^{-1} \left( \frac{x}{\sqrt{1-x2}} \right)$$

2. You must convert from radians to degrees.

```
10 PRINT "YOU ARE COMMANDING
 A GUN":"CREW. HIT WITHIN 10
0 YARDS":"OF THE TARGET"::
20 RANDOMIZE
30 R=INT(40000*RND+20000)
40 PRINT "MAX RANGE";R;"YDS"
50 Z=0
60 S1=0
70 T=INT(R*(.1+.8*RND))
80 S=0
90 GOTO 250
100 PRINT "MIN = 1 DEGREE"
110 GOTO 260
120 PRINT "MAX = 89 DEGREES"
130 GOTO 260
140 PRINT "OVER BY";ABS(E);"
YARDS"
150 GOTO 260
160 PRINT "SHORT BY";ABS(E);
"YARDS"
170 GOTO 260
180 PRINT "TARGET DESTROYED!
"
190 PRINT S;"ROUNDS USED"::
200 S1=S1+S
210 IF Z=4 THEN 410
220 Z=Z+1
230 PRINT "THE FORWARD OBSER
VER HAS":"SIGHTED MORE ENEMY
 ACTIVITY"::
240 GOTO 70
250 PRINT "TARGET AT";T;"YAR
DS"
260 INPUT "GUN ELEVATION =":
B
270 IF B>89 THEN 120
280 IF B<1 THEN 100
290 S=S+1
300 IF S<6 THEN 340
310 PRINT TAB(12);"BOOM!":TA
B(5);"BOOM!":TAB(15);"BOOM!"
:"    BOOM!"::
320 PRINT "ENEMY GOT YOU FIR
ST."
330 GOTO 450
340 B2=.035*B
350 I=R*SIN(B2)
360 X=T-I
370 E=INT(X)
380 IF ABS(E)<100 THEN 180
390 IF E>100 THEN 160
400 GOTO 140
410 PRINT ::"TOTAL ROUNDS US
ED =";S1
420 IF S1>18 THEN 450
430 PRINT "NICE SHOOTING!"
440 GOTO 460
450 PRINT "GO BACK TO FT. SI
LL FOR A":"REFRESHER TRAININ
G COURSE."
460 PRINT
470 INPUT "TRY AGAIN (Y,N) "
:Z$
480 IF Z$="Y" THEN 30
490 PRINT "OK. RETURN TO BAS
E CAMP."
```

```
>RUN
YOU ARE COMMANDING A GUN
CREW. HIT WITHIN 100 YARDS
OF THE TARGET

MAX RANGE 41167 YDS
TARGET AT 17004 YARDS
GUN ELEVATION =77
OVER BY 776 YARDS
GUN ELEVATION =78
SHORT BY 534 YARDS
GUN ELEVATION =77.6
TARGET DESTROYED!
 3 ROUNDS USED

THE FORWARD OBSERVER HAS
SIGHTED MORE ENEMY ACTIVITY

TARGET AT 21714 YARDS
GUN ELEVATION =73
OVER BY 1074 YARDS
GUN ELEVATION =74
SHORT BY 140 YARDS
GUN ELEVATION =73.9
TARGET DESTROYED!
 3 ROUNDS USED

THE FORWARD OBSERVER HAS
SIGHTED MORE ENEMY ACTIVITY

TARGET AT 16942 YARDS
GUN ELEVATION =77.6
TARGET DESTROYED!
 1 ROUNDS USED

THE FORWARD OBSERVER HAS
SIGHTED MORE ENEMY ACTIVITY

TARGET AT 12531 YARDS
GUN ELEVATION =82
SHORT BY 1487 YARDS
GUN ELEVATION =81
SHORT BY 106 YARDS
GUN ELEVATION =80.9
TARGET DESTROYED!
 3 ROUNDS USED

THE FORWARD OBSERVER HAS
SIGHTED MORE ENEMY ACTIVITY

TARGET AT 22631 YARDS
GUN ELEVATION =73.1
TARGET DESTROYED!
 1 ROUNDS USED


TOTAL ROUNDS USED = 11
NICE SHOOTING!

TRY AGAIN (Y,N) N
OK. RETURN TO BASE CAMP.

** DONE **
```



*"I've just programmed our computer to give surprise birthday parties."*

# 9

# Potpourri

Here are five programs that didn't seem to fit anywhere else. The first and the last are games, although you may come to think of the lunar landing simulation as a game also. One is a nifty simulation of smog, and the other calculates depreciation by three different methods.

# Number Guessing Game

Here is a computer program that plays the popular number guessing game. In it the computer picks a secret number between 1 and 100. You attempt to guess that number in as few tries as possible.

There are many ways to go about guessing the secret number. Let some of your friends play this game and see what approaches they use to find the secret number. One approach is to start with a guess of 10. If this is too low, increase each guess by 10 until the computer says that a guess is too high. When this point is reached, start from the previous guess and increase each guess by 1. This is the method used to solve some of the problems in the chapter on convergence.

Another approach is to try to bracket the number between upper and lower limits and reduce the limits by steps until the number is finally found. Two of the convergence programs used this approach.

Is one of these the best way? Well, these methods are not bad, particularly compared to starting with 1 and simply counting to 100 until the solution is found. But there is a better way. It is known as binary search.

This technique involves dividing the search domain, in this case 1 to 100, in half, and then in half again, and so on until the secret number is found. Play the game many times using different approaches. In the long run you should find that the binary search approach is the most efficient.

In Line 210, the program contains the statement that "you should not need more than 7 guesses." Why? If the secret number was between 1 and 128, what is the maximum number of guesses that would be necessary to find it? What if the number range were 1 to 130; then what would be the maximum number of guesses?

Revise the program to choose a secret number between 1 and 10,000. Now the upper limit is 100 times the 1 to 100 game here which requires a maximum of seven guesses to find the secret number; how many guesses will now be required?

Can you write a program in which the roles of the computer and player are reversed? In other words, the computer will try to guess your secret number between 1 and some upper limit. After each guess, you enter L for low, H for high, or C for correct. Can you write this program so the computer can tell if you are cheating, i.e., giving it inconsistent clues?

```
10 RANDOMIZE
20 PRINT "I HAVE CHOSEN A SE
CRET"::"INTEGER BETWEEN 1 AND
 100"::"TRY TO GUESS WHAT IT
IS."::"I WILL GIVE YOU CLUES"
30 PRINT
40 C=0
50 N=INT(RND*100)+1
60 C=C+1
70 INPUT "YOUR GUESS =":G
80 IF N=G THEN 140
90 IF G>N THEN 120
100 PRINT "TOO LOW"
110 GOTO 60
120 PRINT "TOO HIGH"
130 GOTO 60
140 PRINT "CORRECT IN";C;"GU
ESSES"
150 IF C>3 THEN 180
160 PRINT "YOU WERE LUCKY!"
170 GOTO 220
180 IF C>7 THEN 210
190 PRINT "GOOD JOB."
200 GOTO 220
210 PRINT "BUT YOU SHOULDN'T
 NEED MORE":"THAN 7 GUESSES.
"
220 PRINT "PLAY AGAIN (Y OR
N)";
230 INPUT A$
240 PRINT
250 IF A$="Y" THEN 20
260 STOP

>RUN
 I HAVE CHOSEN A SECRET
 INTEGER BETWEEN 1 AND 100
 TRY TO GUESS WHAT IT IS.
 I WILL GIVE YOU CLUES

 YOUR GUESS =50
 TOO LOW
 YOUR GUESS =75
 TOO LOW
 YOUR GUESS =87
 TOO LOW
 YOUR GUESS =93
 TOO LOW
 YOUR GUESS =97
 CORRECT IN 5 GUESSES
 GOOD JOB.
 PLAY AGAIN (Y OR N)? N

 ** DONE **
```

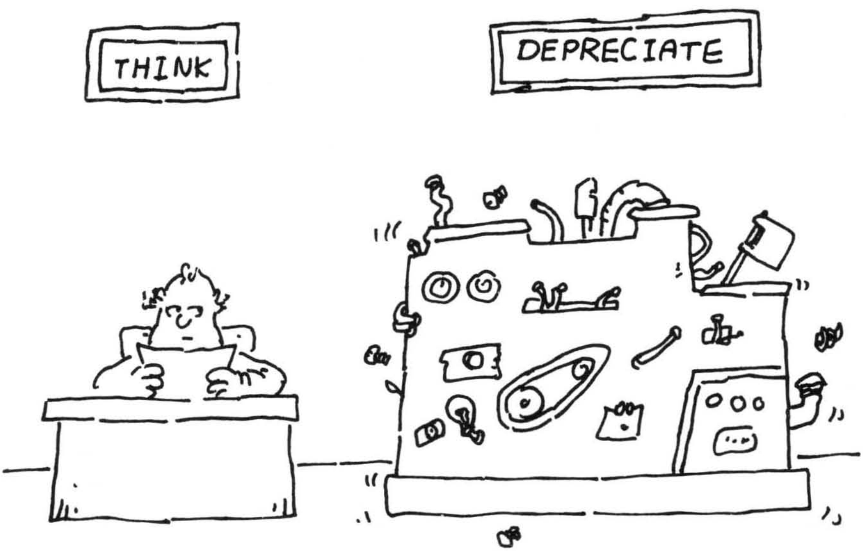
# Depreciation—Three Methods

This program shows how a piece of capital equipment depreciates according to three commonly used methods of depreciation: straight line, sum of the year digits, and double declining.

The program asks for the original cost of the item, its expected life in years (the period of time over which it is to be depreciated), and its expected scrap (or sale) value at the end of that time. A table showing the annual depreciation for each of the three methods is then displayed.

```
10 PRINT "PROGRAM CALCULATES
 DEPRE-":"CIATION BY THREE M
ETHODS":" "
20 INPUT "ORIGINAL COST =":C
30 INPUT "LIFE OF ITEM (YEAR
S) =":L
40 INPUT "SCRAP VALUE =":S
50 PRINT
60 PRINT "    STRAIGHT SUM OF
   DOUBLE":"YEAR LINE    DIGIT
S  DECLINING":" "
70 V=C-S
80 D1=V/L
90 Y=((L+1)/2)*L
100 Z=L
110 FOR X=1 TO L
120 D2=V*(Z/Y)
130 Z=Z-1
140 D3=2*C/L
150 C=C-D3
160 PRINT X;TAB(4);
170 Q=D1
180 GOSUB 280
190 PRINT TAB(12);
200 Q=D2
210 GOSUB 280
220 Q=D3
230 PRINT TAB(21);
240 GOSUB 280
250 PRINT
260 NEXT X
270 STOP
280 Q=INT(Q*100+.5)/100
290 IF Q>1000 THEN 310
300 PRINT " ";
310 IF Q>100 THEN 330
320 PRINT " ";
330 IF Q>10 THEN 350
340 PRINT " ";
350 PRINT Q;
360 RETURN
```

```
PROGRAM CALCULATES DEPRE-
CIATION BY THREE METHODS

ORIGINAL COST =8000
LIFE OF ITEM (YEARS) =8
SCRAP VALUE =500

   STRAIGHT SUM OF  DOUBLE
YEAR LINE   DIGITS DECLINING

 1    937.5  1666.67  2000
 2    937.5  1458.33  1500
 3    937.5  1250     1125
 4    937.5  1041.67   843.75
 5    937.5   833.33   632.81
 6    937.5   625      474.61
 7    937.5   416.67   355.96
 8    937.5   208.33   266.97

** DONE **
```



# Smog Simulation

This program is an adaptation of the smog model originally written by Herbert Peckham. The model assumes that vehicular traffic is the sole producer of smog, a somewhat poor assumption. It also assumes that most automobile traffic occurs during the daylight hours and that traffic volume is very low (actually, zero) at night. The smog generated by the cars is dissipated by atmospheric conditions which vary depending upon sunlight, temperature, and weather. All of these conditions may be specified by the user.

The model could be improved significantly by taking into account other sources of smog, by varying vehicular traffic according to the hour of the day, and by allowing daily variation of weather factors. Nevertheless, even in this rudimentary form it is interesting and instructive.

A plot of the smog level is produced in Statement 370. Under some conditions, the smog level reaches a value that cannot be plotted because it is greater than the width of the screen (and printer). For runs with conditions like this, you might want to delete the plotting routine. A more elegant solution would be to estimate the maximum value of the smog level from the input factors and calculate an appropriate plotting multiplication factor.

```
10 PRINT "SMOG CITY SIMULATI
ON":" "
20 T=7
30 PRINT "CITY     CARS PER RO
AD MILE":"LOS ANGELES   200"
:"DETROIT          100":"TULSA
           25":"SMOG CITY    ";
40 INPUT C
50 PRINT
60 PRINT "     SMOG GENERATIO
N CONSTANT":"1950 AUTO    .00
5":"1975 AUTO    .001":"BUS
         .010":"SMOG CITY  ";
70 INPUT L1
80 PRINT
90 PRINT "DAYTIME SMOG BREAK
DOWN RATE":"IN PERCENT PER H
OUR"
100 PRINT "  CLEAR       .02"
:"  CLOUDY      .01":"  SMOG
CITY";
110 INPUT R1
120 PRINT
130 PRINT "NIGHTTIME SMOG BR
EAKDOWN":"RATE (% PER HOUR)"
140 PRINT "  HOT         .05"
:"  COOL        .10":"  SMOG
CITY";
150 INPUT R2
160 PRINT
170 PRINT "SMOG DISPERSION (
% PER HOUR)"
180 PRINT "HIGH WIND AND RAI
N    1.00":"NO WIND AND DRY
  0.01":"SMOG CITY
";
190 INPUT R3
200 PRINT " ":"HOUR SMOG LEV
EL       PLOT"
210 R=R1
220 K1=L1
230 T1=INT((T-6)/12)
240 IF T1/2=INT(T1/2)THEN 27
0
250 K1=0
260 R=R2
270 S=S+K1*C*10-R*S-R3*S
280 IF S>0 THEN 290
290 T=T+1
300 T3=INT(T/12)
310 T2=T-T3*12+1
320 X=INT(1000*S+.5)/1000
330 IF T3/2=INT(T3/2)THEN 39
0
340 PRINT T2;"PM";TAB(7);X;
350 IF X<14 THEN 370
360 X=14
370 PRINT TAB(13+X);"*"
380 GOTO 210
390 PRINT T2;"AM";TAB(7);X;
400 GOTO 370
```

```
>RUN
 SMOG CITY SIMULATION

 CITY     CARS PER ROAD MILE
 LOS ANGELES    200
 DETROIT        100
 TULSA           25
 SMOG CITY    ? 100

      SMOG GENERATION CONSTANT
 1950 AUTO    .005
 1975 AUTO    .001
 BUS          .010
 SMOG CITY ?  .001

 DAYTIME SMOG BREAKDOWN RATE
 IN PERCENT PER HOUR
   CLEAR        .02
   CLOUDY       .01
   SMOG CITY?   .01

 NIGHTTIME SMOG BREAKDOWN
 RATE (% PER HOUR)
   HOT          .05
   COOL         .10
   SMOG CITY?   .10

 SMOG DISPERSION (% PER HOUR)
 HIGH WIND AND RAIN   1.00
 NO WIND AND DRY      0.01
 SMOG CITY          ? .05

 HOUR SMOG LEVEL        PLOT
  9 AM  1      *
  10 AM 1.94    *
  11 AM 2.824    *
  12 AM 3.654     *
  1 PM  4.435     *
  2 PM  5.169      *
  3 PM  5.859       *
  4 PM  6.507        *
  5 PM  7.117        *
  6 PM  7.69          *
  7 PM  8.228         *
  8 PM  6.994        *
  9 PM  5.945       *
  10 PM 5.053      *
  11 PM 4.295     *
  12 PM 3.651     *
  1 AM  3.103    *
  2 AM  2.638    *
  3 AM  2.242   *
  4 AM  1.906   *
  5 AM  1.62    *
  6 AM  1.377  *
  7 AM  1.17   *
  8 AM  2.1     *
  9 AM  2.974    *

  9 PM   13.538                *
  10 PM  11.507              *
  11 PM  9.781             *
  12 PM  8.314           *
  1 AM   7.067          *
  2 AM   6.007         *
```

# Lunar Lander Simulation

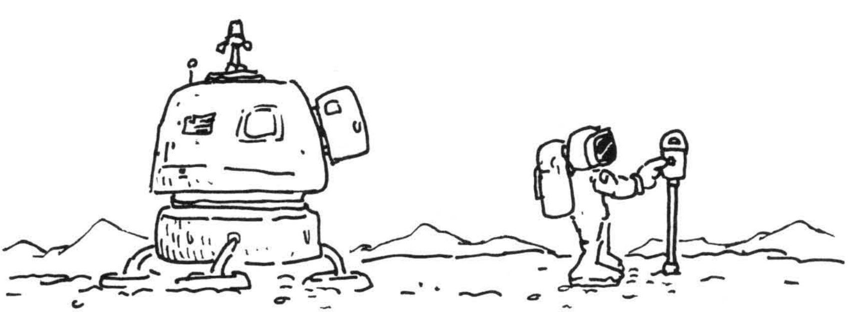
This program is one of the most popular computer simulations around. It is available in many versions; this one is adapted from the original program written in 1969.

The program represents an exact simulation of an Apollo lunar landing module during the final descent. This portion of the descent would normally be controlled by the on-board computer backed up by another computer in the lunar orbiter, and still another computer on Earth. However, to exercise your knowledge of physics and to make an interesting game, we will assume that all three computers have had a simultaneous malfunction. Hence, it is up to you to land the spacecraft safely.

To make a soft landing, you may change the burn rate of the retro rockets every ten seconds. You have a choice of not firing at all (burn rate of 0) or of firing at a fuel rate of between 8 and 200 pounds per second. Engine ignition occurs at 8 pounds, hence values between 1 and 7 pounds are not possible. You have 16,500 pounds of fuel. This is 500 pounds more than an actual LEM has, which will give you a little margin for error. When you get proficient, change Statement 80 to N = 16000 to simulate the real thing more closely. The capsule weight is 33,000 pounds.

Not that this is the way to come in, but if you did not fire the rockets at all, the estimated time for a free fall descent is 120 seconds to impact (and a huge splat).

Good luck!



```
10 PRINT "COMPUTER MALFUNCTI
ON!":"ASSUME MANUAL CONTROL
OF":"LUNAR CAPSULE...":
20 PRINT "CAPSULE WEIGHT 32,
500 LBS.":"AVAILABLE FUEL 16
,500 LBS.":
30 PRINT "SET RETRO ROCKET B
URN RATE":"EVERY 10 SECONDS
TO 0 (ZERO)":"OR VALUE BETWE
EN 8 AND 200":"LBS PER SEC":
40 PRINT ::" SEC MILES MPH
 FUEL    BURN"::
50 A=120
60 V=1
70 M=33000
80 N=16500
90 G=.001
100 Z=1.8
110 P=INT(3600*V+.5)
120 IF P>=1 THEN 140
130 P=INT((3600*V+.5)*1000)/
1000
140 IF A<10 THEN 170
150 D=INT(A+.5)
160 GOTO 180
170 D=(INT(1000*A+.5))/1000
180 PRINT L;D;TAB(11);P;TAB(
17);M-N;
190 PRINT TAB(24);
200 INPUT K
210 IF K>200 THEN 250
220 IF K>7 THEN 270
230 IF K=0 THEN 270
240 T=10
250 PRINT "BAD VALUE--AGAIN
PLEASE";
260 GOTO 200
270 T=10
280 IF M-N<.001 THEN 390
290 IF T<.001 THEN 110
300 S=T
310 IF M>N+S*K THEN 330
320 S=(M-N)/K
330 GOSUB 770
340 IF I<=0 THEN 630
350 IF V<=0 THEN 370
360 IF J<0 THEN 690
370 GOSUB 570
380 GOTO 280
390 PRINT "OUT OF FUEL AT";L
;"SEC"
400 S=(-V+SQR(V*V+2*A*G))/G
410 V=V+G*S
420 L=L+S
430 W=3600*V
440 PRINT "ON MOON AT";INT(L
);"SEC; IMPACT":"VELOCITY ="
;INT(100*W+.5)/100;"MPH."
450 IF W>1.2 THEN 480
460 PRINT "PERFECT LANDING!"
470 STOP
480 IF W>10 THEN 510
490 PRINT "OOOOMPH! THAT'S R
OUGH ON":"THE SHOCK ABSORBER
S"
500 STOP
```

```
510 IF W>60 THEN 540
520 PRINT "SEVERE CRAFT DAMA
GE. YOU'RE":"STRANDED UNTIL
HELP COMES"
530 STOP
540 PRINT "SORRY, BUT NO SUR
VIVORS;":"YOU BLEW IT. IN FA
CT, YOU":"BLASTED A NEW LUNA
R CRATER"
550 PRINT INT(W*.278);"FEET
DEEP!"
560 STOP
570 L=L+S
580 T=T-S
590 M=M-S*K
600 A=I
610 V=J
620 RETURN
630 IF S<.005 THEN 430
640 D=V+SQR(V*V+2*A*(G-Z*K/M
))
650 S=2*A/D
660 GOSUB 770
670 GOSUB 570
680 GOTO 630
690 W=(1-M*G/(Z*K))/2
700 S=M*V/(Z*K*(W+SQR(W*W+V/
Z)))+.05
710 GOSUB 770
720 IF I<=0 THEN 630
730 GOSUB 570
740 IF J>0 THEN 280
750 IF V>0 THEN 690
760 GOTO 280
770 Q=S*K/M
780 J=V+G*S-Z*Q*(1+Q*(.5+Q*(
1/3+Q*(.25+Q/5))))
790 I=A-G*S*S/2-V*S+Z*S*Q*(.
5+Q*(1/6+Q*(1/12+Q/20)))
800 RETURN
```

```
COMPUTER MALFUNCTION!
ASSUME MANUAL CONTROL OF
LUNAR CAPSULE...

CAPSULE WEIGHT 32,500 LBS.
AVAILABLE FUEL 16,500 LBS.

SET RETRO ROCKET BURN RATE
EVERY 10 SECONDS TO 0 (ZERO)
OR VALUE BETWEEN 8 AND 200
LBS PER SEC

 SEC MILES MPH    FUEL    BURN

 0   120    3600   16500 ? 0
 10   110   3636   16500 ? 0
 20   100   3672   16500 ? 0
 30    90   3708   16500 ? 0
 40    79   3744   16500 ? 0
 50    69   3780   16500 ? 0
 60    58   3816   16500 ? 200
```

```
 70   48      3447   14500  ?  200
 80   39      3051   12500  ?  200
 90   31      2624   10500  ?  0
 100   24     2660   10500  ?  99
 110   17     2454   9510   ?  99
 120   10     2238   8520   ?  99
 130   4.321
              2013   7530   ?  99
ON MOON AT 138 SEC; IMPACT
VELOCITY = 1821.47 MPH.
SORRY, BUT NO SURVIVORS;
YOU BLEW IT. IN FACT, YOU
BLASTED A NEW LUNAR CRATER
 506 FEET DEEP!

** DONE **




COMPUTER MALFUNCTION!
ASSUME MANUAL CONTROL OF
LUNAR CAPSULE...

CAPSULE WEIGHT 32,500 LBS.
AVAILABLE FUEL 16,500 LBS.

SET RETRO ROCKET BURN RATE
EVERY 10 SECONDS TO 0 (ZERO)
OR VALUE BETWEEN 8 AND 200
LBS PER SEC


 SEC MILES MPH    FUEL    BURN

 0   120      3600   16500  ?  0
 10   110     3636   16500  ?  0
 20   100     3672   16500  ?  0
 30   90      3708   16500  ?  0
 40   79      3744   16500  ?  0
 50   69      3780   16500  ?  0
 60   58      3816   16500  ?  0
 70   48      3852   16500  ?  200
 80   37      3483   14500  ?  200
 90   28      3087   12500  ?  200
 100   20     2660   10500  ?  200
 110   13     2197   8500   ?  200
 120   8.07   1693   6500   ?  200
 130   4.1251139     4500   ?  200
 140   1.796527      2500   ?  100
 150   .766   212    1500   ?  50
 160   .378   66     1000   ?  19
 170   .244   31     810    ?  13
 180   .176   18     680    ?  12
 190   .139   9      560    ?  10
 200   .117   7      460    ?  8
 210   .091   12     380    ?  9
 220   .055   13     290    ?  10
 230   .023   11     190    ?  10
ON MOON AT 238 SEC; IMPACT
VELOCITY = 7.98 MPH.
OOOOMPH! THAT'S ROUGH ON
THE SHOCK ABSORBERS
```

# Hammurabi

Hammurabi is one of the all-time favorite computer games. On the one hand, it may be considered a game, but on the other it is an intriguing simulation of barter and management.

Hammurabi is your servant as you try to manage the ancient city-state of Sumeria. The economy of the city-state revolves around just one thing—the annual crop of grain (probably soybeans).

Each year, you must determine how many bushels of grain you wish to feed to your people (you'll quickly discover how much a person needs to survive), how much you wish to use as seed in planting crops for the coming year, how much you wish to use for the purchase of additional land from your neighboring city-state, and how much you wish to put in storage.

Of course, if you have a bad harvest or if rats overrun your grain storage bins, you may have to sell land in order to get enough grain to keep your people from starving, or to plant the land for the coming year. Unfortunately, disasters always seem to strike, forcing you to sell land, when the price is at an all-time low; but that's not any different from the real world.

Most people start to play this game with noble ambitions. However, before long, they start longing for a plague to trim their growing population. Or they deliberately starve some people to keep things in balance (gosh, maybe these zero population growth people have something, after all!).

Over the years, this game more than any other, has spawned a host of look-alikes, extensions, and modifications. Indeed, several manufacturers have taken my original with no changes whatsoever, put it in a fancy box, and charged a handsome price for it. Accept no imitations! Here is the original game (with the dialog shortened slightly) for you to run on your computer.

If you want to experiment with changes, here are some suggestions. In the existing game, plagues randomly occur 15% of the time; lower this to 10% or 5%. People now require a fixed amount of food; vary this amount slightly from year to year. Permit the construction of a rat-proof grain bin, but this must cost a fair amount. Introduce a mining industry as well as agriculture. How about fishing or tourism? Let your imagination run wild. Experiment! Have fun!

```
10 CALL CLEAR
20 PRINT TAB(9);"HAMMURABI"
30 PRINT "       BY DAVID H.
AHL"::
40 PRINT "TRY TO SUCCESSFULL
Y GOVERN":"ANCIENT SUMERIA F
OR 10 YRS"::
50 D1=0
60 P1=0
70 Z=0
80 P=95
90 S=2800
100 H=3000
110 E=H-S
120 Y=3
130 A=H/Y
140 I=5
150 Q=1
160 D=0
170 PRINT
180 Z=Z+1
190 IF Z=11 THEN 1120
200 PRINT "HAMMURABI: I BEG
TO REPORT,"
210 PRINT "YEAR";Z;",";D;"PE
OPLE STARVED":"AND";I;"WERE
BORN."
220 P=P+I
230 IF Q>0 THEN 260
240 P=INT(P/2)
250 PRINT "A HORRIBLE PLAGUE
 STRUCK!":"HALF THE POPULATI
ON DIED."
260 PRINT "POPULATION IS NOW
";P
270 PRINT "YOU OWN";A;"ACRES
 WHICH"
280 PRINT "YIELDED";Y;"BUSHE
LS PER ACRE."
290 PRINT "RATS ATE";E;"BUSH
ELS. YOU"
300 PRINT "HAVE";S;"BSHLS IN
 STORAGE."::
310 Y=INT(17+RND*10)
320 PRINT "LAND COSTS";Y;"BU
SHELS/ACRE"
330 INPUT "BUY HOW MANY ACRE
S?":Q
340 IF Q<0 THEN 1100
350 IF Y*Q<=S THEN 380
360 GOSUB 1040
370 GOTO 330
380 IF Q=0 THEN 430
390 A=A+Q
400 S=S-Y*Q
410 C=0
420 GOTO 510
430 INPUT "SELL HOW MANY ACR
ES?":Q
440 IF Q<0 THEN 1100
450 IF Q<A THEN 480
460 GOSUB 1060
470 GOTO 430
480 A=A-Q
490 S=S+Y*Q
```

```
500  C=0
510  PRINT
520  INPUT  "BUSHELS  TO  FEED  P
EOPLE?":Q
530  IF  Q<0  THEN  1100
540  REM  !NOT  ENOUGH  GRAIN?
550  IF  Q<=S  THEN  580
560  GOSUB  1040
570  GOTO  510
580  S=S-Q
590  C=1
600  PRINT
610  INPUT  "PLANT  HOW  MANY  AC
RES?":D
620  IF  D=0  THEN  760
630  IF  D<0  THEN  1100
640  REM  !NOT  ENOUGH  LAND?
650  IF  D<=A  THEN  690
660  GOSUB  1060
670  GOTO  610
680  REM  !ENOUGH  SEED  GRAIN?
690  IF  INT(D/2)<=S  THEN  720
700  GOSUB  1040
710  GOTO  610
720  IF  D<10*P  THEN  750
730  PRINT  "YOU  DO  NOT  HAVE  E
NOUGH":"PEOPLE  TO  TEND  THE  C
ROPS."
740  GOTO  610
750  S=S-INT(D/2)
760  GOSUB  1080
770  REM  !A  BOUNTIFUL  HARVEST
780  Y=C
790  H=D*Y
800  E=0
810  GOSUB  1080
820  IF  INT(C/2)<>C/2  THEN  85
0
830  REM  !RATS  ARE  GOING  WILD
840  E=INT(S/C)
850  S=S-E+H
860  GOSUB  1080
870  REM  !LET'S  HAVE  BABIES
880  I=INT(C*(20*A+S)/P/100+1
)
890  REM  !PEOPLE  WELL  FED?
900  C=INT(Q/20)
910  REM  !15%  CHANCE  OF  PLAGU
E
920  Q=INT(10*(2*RND-.3))
930  IF  P<C  THEN  160
940  D=P-C
950  REM  !STARVE  ENOUGH  FOR
          IMPEACHMENT?
960  IF  D>.45*P  THEN  1010
970  P1=((Z-1)*P1+D*100/P)/Z
980  P=C
990  D1=D1+D
1000  GOTO  170
```

```
1010 PRINT ::"YOU STARVED";D
"PEOPLE IN A":"SINGLE YEAR!"
1020 PRINT "   DUE TO THIS E
XTREME":"MISMANAGMENT, YOU H
AVE BEEN":"IMPEACHED AND DEC
LARED THE":"NATIONAL FINK."
1030 GOTO 1260
1040 PRINT "HAMMURABI: THINK
 AGAIN. YOU":"HAVE ONLY";S;"
BSHLS OF GRAIN"
1050 RETURN
1060 PRINT "HAMMURABI: THINK
 AGAIN. YOU":"OWN ONLY";A;"A
CRES."
1070 RETURN
1080 C=INT(5*RND)+1
1090 RETURN
1100 PRINT ::"YOU ARE ASKING
 ME TO DO":"IMPOSSIBLE THING
S. GET":"YOURSELF ANOTHER ST
EWARD."
1110 GOTO 1260
1120 PRINT "HAMMURABI REPORT
S THAT IN 10":"YEARS, YOU ST
ARVED";D1:"PEOPLE. THAT'S";I
NT(100*P1)/100;"% PER YEAR!"
1130 L=A/P
1140 PRINT "YOU STARTED WITH
 10 ACRES":"PER PERSON AND E
NDED WITH";INT(10*L)/10:::
1150 IF P1>33 THEN 1020
1160 IF L<7 THEN 1020
1170 IF P1>10 THEN 1230
1180 IF L<9 THEN 1230
1190 IF P1>3 THEN 1250
1200 IF L<10 THEN 1250
1210 PRINT "YOUR 10-YR PERFO
RMANCE WAS":"FANTASTIC. WHY
DON'T YOU RUN":"FOR GOVERNOR
 OF NEW JERSEY?"
1220 GOTO 1260
1230 PRINT "YOU RULE WITH A
HEAVY HAND.":"THE PEOPLE HAT
E YOUR GUTS":"AND ARE REBELL
ING. BETTER":"GET OUT QUICK!
"
1240 GOTO 1260
1250 PRINT "YOUR MANAGEMENT
WASN'T TOO":"BAD, BUT THERE
IS ROOM FOR":"IMPROVEMENT."
1260 PRINT ::"HAMMURABI: SO
LONG FOR NOW.":: 
1270 END
```

```
              HAMMURABI
          BY DAVID H. AHL

      TRY TO SUCCESSFULLY GOVERN
      ANCIENT SUMERIA FOR 10 YRS


      HAMMURABI: I BEG TO REPORT,
      YEAR 1 , 0 PEOPLE STARVED
      AND 5 WERE BORN.
      POPULATION IS NOW 100
      YOU OWN 1000 ACRES WHICH
      YIELDED 3 BUSHELS PER ACRE.
      RATS ATE 200 BUSHELS. YOU
      HAVE 2800 BSHLS IN STORAGE.

      LAND COSTS 22 BUSHELS/ACRE
      BUY HOW MANY ACRES? 0
      SELL HOW MANY ACRES? 0

      BUSHELS TO FEED PEOPLE?2000

      PLANT HOW MANY ACRES?999

      HAMMURABI: I BEG TO REPORT,
      YEAR 2 , 0 PEOPLE STARVED
      AND 5 WERE BORN.
      POPULATION IS NOW 105
      YOU OWN 1000 ACRES WHICH
      YIELDED 2 BUSHELS PER ACRE.
      RATS ATE 0 BUSHELS. YOU
      HAVE 2299 BSHLS IN STORAGE.

      LAND COSTS 22 BUSHELS/ACRE
      BUY HOW MANY ACRES? 0
      SELL HOW MANY ACRES? 0

      BUSHELS TO FEED PEOPLE?2010

      PLANT HOW MANY ACRES?999
      HAMMURABI: THINK AGAIN. YOU
      HAVE ONLY 289 BSHLS OF GRAIN
      PLANT HOW MANY ACRES?577

      HAMMURABI: I BEG TO REPORT,
      YEAR 3 , 5 PEOPLE STARVED
      AND 9 WERE BORN.
      A HORRIBLE PLAGUE STRUCK!
      HALF THE POPULATION DIED.
      POPULATION IS NOW 54
      YOU OWN 1000 ACRES WHICH
      YIELDED 2 BUSHELS PER ACRE.
      RATS ATE 0 BUSHELS. YOU
      HAVE 1155 BSHLS IN STORAGE.

      LAND COSTS 26 BUSHELS/ACRE
      BUY HOW MANY ACRES? 0
      SELL HOW MANY ACRES? 50

      BUSHELS TO FEED PEOPLE?1080

      PLANT HOW MANY ACRES?530
```

```
HAMMURABI:  I BEG TO REPORT,
YEAR 4 , 0 PEOPLE STARVED
AND 9 WERE BORN.
A HORRIBLE PLAGUE STRUCK!
HALF THE POPULATION DIED.
POPULATION IS NOW 31
YOU OWN 950 ACRES WHICH
YIELDED 3 BUSHELS PER ACRE.
RATS ATE 0 BUSHELS. YOU
HAVE 2700 BSHLS IN STORAGE.

LAND COSTS 25 BUSHELS/ACRE
BUY HOW MANY ACRES? 0
SELL HOW MANY ACRES? 50

BUSHELS TO FEED PEOPLE?620

PLANT HOW MANY ACRES?309

HAMMURABI:  I BEG TO REPORT,
YEAR 5 , 0 PEOPLE STARVED
AND 29 WERE BORN.
POPULATION IS NOW 60
YOU OWN 900 ACRES WHICH
YIELDED 4 BUSHELS PER ACRE.
RATS ATE 0 BUSHELS. YOU
HAVE 4412 BSHLS IN STORAGE.

LAND COSTS 24 BUSHELS/ACRE
BUY HOW MANY ACRES? 0
SELL HOW MANY ACRES? 0

BUSHELS TO FEED PEOPLE?1200

PLANT HOW MANY ACRES?599

HAMMURABI:  I BEG TO REPORT,
YEAR 6 , 0 PEOPLE STARVED
AND 12 WERE BORN.
A HORRIBLE PLAGUE STRUCK!
HALF THE POPULATION DIED.
POPULATION IS NOW 36
YOU OWN 900 ACRES WHICH
YIELDED 5 BUSHELS PER ACRE.
RATS ATE 728 BUSHELS. YOU
HAVE 5180 BSHLS IN STORAGE.

LAND COSTS 20 BUSHELS/ACRE
BUY HOW MANY ACRES? 100

BUSHELS TO FEED PEOPLE?720

PLANT HOW MANY ACRES?359

HAMMURABI:  I BEG TO REPORT,
YEAR 7 , 0 PEOPLE STARVED
AND 7 WERE BORN.
POPULATION IS NOW 43
YOU OWN 1000 ACRES WHICH
YIELDED 4 BUSHELS PER ACRE.
RATS ATE 0 BUSHELS. YOU
HAVE 3717 BSHLS IN STORAGE.

LAND COSTS 23 BUSHELS/ACRE
BUY HOW MANY ACRES? 0
SELL HOW MANY ACRES? 0
```

```
BUSHELS TO FEED PEOPLE?860

PLANT HOW MANY ACRES?429

HAMMURABI: I BEG TO REPORT,
YEAR 8 , 0 PEOPLE STARVED
AND 12 WERE BORN.
POPULATION IS NOW 55
YOU OWN 1000 ACRES WHICH
YIELDED 4 BUSHELS PER ACRE.
RATS ATE 0 BUSHELS. YOU
HAVE 4359 BSHLS IN STORAGE.

LAND COSTS 25 BUSHELS/ACRE
BUY HOW MANY ACRES? 0
SELL HOW MANY ACRES? 50

BUSHELS TO FEED PEOPLE?1100

PLANT HOW MANY ACRES?549

HAMMURABI: I BEG TO REPORT,
YEAR 9 , 0 PEOPLE STARVED
AND 17 WERE BORN.
A HORRIBLE PLAGUE STRUCK!
HALF THE POPULATION DIED.
POPULATION IS NOW 36
YOU OWN 950 ACRES WHICH
YIELDED 1 BUSHELS PER ACRE.
RATS ATE 1058 BUSHELS. YOU
HAVE 3726 BSHLS IN STORAGE.

HAMMURABI: I BEG TO REPORT,
YEAR 10 , 0 PEOPLE STARVED
AND 24 WERE BORN.
POPULATION IS NOW 60
YOU OWN 950 ACRES WHICH
YIELDED 3 BUSHELS PER ACRE.
RATS ATE 1413 BUSHELS. YOU
HAVE 2491 BSHLS IN STORAGE.

LAND COSTS 23 BUSHELS/ACRE
BUY HOW MANY ACRES? 0
SELL HOW MANY ACRES? 0

BUSHELS TO FEED PEOPLE?1200

PLANT HOW MANY ACRES?599

HAMMURABI REPORTS THAT IN 10
YEARS, YOU STARVED 5
PEOPLE. THAT'S .47
% PER YEAR!
YOU STARTED WITH 10 ACRES
PER PERSON AND ENDED WITH
 12.6

YOUR 10-YR PERFORMANCE WAS
FANTASTIC. WHY DON'T YOU RUN
FOR GOVERNOR OF NEW JERSEY?

HAMMURABI: SO LONG FOR NOW.
```


*"Rats! A bacterium just ate the new micro-mini computer."*

# References

Magazines referred to in the text include:

• *99'er.* This magazine focuses on Texas Instruments home computers exclusively. It carries articles and reviews of peripherals, software and other accessories. Write for current subscription information:

*99'er*
Box 5537
Eugene, OR 97405.

• *Creative Computing.* This is the leading magazine of software and applications for all small computers. It carries articles, tutorials, how-to applications, and extensive in-depth evaluations.

Books referred to in the text include:

• *Computers in Mathematics: A Sourcebook of Ideas.* Hundreds of classroom-tested ideas for using computers to learn about mathematics.

• *Computers in Science and Social Studies.* Scores of simulation programs in biology, ecology, physics and management of real world systems.

All of these books and magazines (except *99'er*) are available from *Creative Computing.* Write or call for the current price:

Creative Computing
39 E. Hanover Ave.
Morris Plains, NJ 07950
(800) 631-8112
In NJ (201) 540-0445

THE TEXAS INSTRUMENTS HOME COMPUTER

$8.95

# IDEA BOOK

This **Ideabook** contains dozens of ways to make the most out of your computer for solving practical, everyday problems. The 50 ready-to-run programs demonstrate scores of different techniques for solving problems in mathematics, science and business.

The ten chapters deal with solving problems by formulas and repetitive trials, convergence, recursion, compounding, probability, geometry, science, simulations, and drill and practice.

Some of the problems demonstrate the capabilities of the computer; others identify its shortcomings. It is important to be familiar with both the strengths and weaknesses of your tools so you can recognize the types of jobs for which they are suitable.

The author, David H. Ahl, has been involved with the use of computers since 1957. He is the author of 16 books and is the founder and editorial director of Creative Computing, SYNC and Video & Arcade Games magazines.

CREATIVE COMPUTING PRESS

ISBN # 0-916688-51-8